VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2020 Bc. Jiří Zahradník

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

TESTOVÁNÍ ZRANITELNOSTÍ V PRŮMYSLOVÝCH SÍTÍCH
VULNERABILITIES ASSESSMENT FOR INDUSTRIAL PROTOCOLS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Jiří Zahradník**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Petr Blažek**
SUPERVISOR

BRNO 2020

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Informační bezpečnost**

Ústav telekomunikací

Student: Bc. Jiří Zahradník ***ID:*** 165553
Ročník: 2 ***Akademický rok:*** 2019/20

NÁZEV TÉMATU:

Testování zranitelností v průmyslových sítích

POKYNY PRO VYPRACOVÁNÍ:

Práce je zaměřená na analýzu a simulování zranitelností, které se vyskytují v průmyslových sítích. Student provede analýzu průmyslových protokolů (např. IEC 61850, IEC 60870, DNP3 a další) a jejich zranitelností (útoky, bezpečnostní nedostatky protokolů atd.). Bude zprovozněna jednoduchá ICS/SCADA síť s alespoň dvěma vybranými protokoly. Následně bude proveden návrh metodiky pro ověření bezpečnosti pomocí simulačního nástroje (např. Kali Linux). Bude provedeno ověření bezpečnosti jednotlivých komunikačních protokolů a zařízení, společně se simulací zranitelností. Výstupem práce bude alespoň sedm realizovaných a ověřených zranitelností na vytvořené infrastruktuře. Dílčím výsledkem bude návrh a ověření mitigačních opatření pro zvýšení zabezpečení sítě.

DOPORUČENÁ LITERATURA:

[1] KRUTZ, Ronald L. Securing SCADA systems. John Wiley & Sons, 2005.

[2] HERTZOG, Raphael a O'GORMAN, Jim. Kali Linux Revealed: Mastering the Penetration Testing Distribution. Offsec Press, 2017.

Termín zadání: 3.2.2020 ***Termín odevzdání:*** 2.6.2020

Vedoucí práce: Ing. Petr Blažek

prof. Ing. Jiří Mišurec, CSc.
předseda oborové rady

ABSTRAKT

Práce simuluje vybrané zranitelnosti standardu IEC 61850 a následně řeší mitigační opatření pro zvolené zranitelnosti.
Autor simuloval zranitelnosti protokolu GOOSE, útok na NTP a útok na MMS klienta. Konkrétně se jedná o GOOSE stNum, GOOSE semantic, GOOSE test bit, GOOSE replay, GOOSE flood, NTP spoofing a MMS password capture.
Útoky na protokoly GOOSE a MMS byly úspěšné, útok na NTP byl pouze částečně úspěšný, kdy došlo k potvrzení přijatého času, ale ne k jeho převzetí. Autor následně navrhnul možná mitigační opatření. Byl také vytvořen automatizační nástroj pro testování daných zranitelností, parser pro protokol GOOSE a přenositelný parser konfiguračních souborů.
Výsledky práce umožňují implementovat rozsáhlejší nástroj pro penetrační testování průmyslových sítí, stejně jako umožňují implementovat daná mitigační opatření.

KLÍČOVÁ SLOVA

GOOSE, NTP, MMS, IEC 61850, IEC 62351, GOOSE stNum, GOOSE replay, GOOSE flood, GOOSE semantic, NTP spoofing, MMS password capture, mitigační opatření

ABSTRACT

Thesis deals with testing of selected vulnerabilities from the IEC 61850 standard and following design of mitigation measures for selected vulnerabilities.
Author simulated vulnerabilities of the GOOSE protocol, NTP attack and attack on a MMS client. Those attacks were GOOSE stNum, GOOSE semantic, GOOSE test bit, GOOSE replay, GOOSE flood, NTP spoofing and MMS password capture.
Attacks on protocols GOOSE and MMS were successful, attack on NTP was only partially successful since the device confirmed receiving spoofed time, however it did not change it's inner clock. Author then designed possible mitigation measures. Tool for automatic testing of selected vulnerabilities, parser for the GOOSE protocol and lightweight multiplatform parser for configuration files were created as well.
The outcome of this thesis allows the implementation of lager scale tool for penetration testing of industrial networks as well as it allows implementation of discussed mitigation measures.

KEYWORDS

GOOSE, NTP, MMS, IEC 61850, IEC 62351, GOOSE stNum, GOOSE replay, GOOSE flood, GOOSE semantic, NTP spoofing, MMS password capture, cyberattack mitigation

ZAHRADNÍK, Jiří. *Testování zranitelností v průmyslových sítích*. Brno, 2020, 81 s. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce: Ing. Petr Blažek

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Testování zranitelností v průmyslových sítích“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora

PODĚKOVÁNÍ

Rád bych poděkoval svému vedoucímu diplomové práce panu Ing. Petru Blažkovi za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci. Dále chci poděkovat svým rodičům za veškerou podporu a trpělivost, kterou mi poskytli, své přítelkyni, a svým kamarádům, kteří mě podrželi, když jsem potřeboval.

Obsah

Úvod **12**

1 Teoretická část studentské práce **13**
1.1 Kritická infrastruktura . . . 13
1.1.1 Energetická kritická infrastruktura . . . 13
1.1.2 Následky poškození kritické infrastruktury . . . 14
1.2 ICS Kill Chain . . . 17
1.2.1 Fáze 1 . . . 17
1.2.2 Fáze 2 . . . 18
1.3 Známý malware . . . 19
1.3.1 BlackEnergy . . . 19
1.3.2 Industroyer . . . 21
1.3.3 Stuxnet . . . 23
1.4 Známé útoky . . . 24
1.4.1 Útok na německé ocelárny v roce 2014 . . . 24
1.4.2 Ivano-Frankivsk . . . 25

2 Výsledky diplomové práce **27**
2.1 IEC 61850 . . . 27
2.1.1 Informační model . . . 27
2.1.2 Protokol GOOSE . . . 28
2.1.3 Protokol MMS . . . 30
2.2 IEC 62351 . . . 31
2.3 Model testovací infrastruktury . . . 31
2.3.1 Fyzická infrastruktura . . . 32
2.3.2 Virtualizovaná infrastruktura . . . 32
2.4 Útoky . . . 33
2.4.1 Útok na simulační bit protokolu GOOSE . . . 34
2.4.2 NTP Spoofing . . . 38
2.4.3 GOOSE replay . . . 41
2.4.4 GOOSE flood . . . 46
2.4.5 GOOSE stNum . . . 51
2.4.6 GOOSE semantic . . . 54
2.4.7 MMS password capture . . . 59
2.5 Testovací nástroje pro usnadnění práce a konfigurace útoků . . . 64
2.5.1 Parser konfiguračních souborů . . . 64
2.5.2 Parser protokolu GOOSE . . . 66

2.5.3 Automatizační nástroj 68
2.6 Návrh mitigačních opatření 70

Závěr **76**

Literatura **77**

Seznam symbolů, veličin a zkratek **80**

Seznam obrázků

1.1 Rozvodná síť České republiky. 14
1.2 Zasažené státy při severovýchodním blackoutu v roce 2003. 16
1.3 Rozdělení ICS KC a mapování útoku na Ivano-Frankivsk v roce 2015. 17
1.4 Zjednodušené schéma Industroyeru. 22
1.5 Útok na německé ocelárny v roce 2014. 25
2.1 Informační model standardu IEC61850. 28
2.2 Zapouzdření protokolu GOOSE. 29
2.3 Klient-server model MMS. 30
2.4 Model fyzické infrastruktury. 32
2.5 Model virtualizované infrastruktury. 33
2.6 Originální pakety (vlevo) a podvržené pakety (vpravo). 37
2.7 Model testovací infrastruktury pro NTP spoofing. 38
2.8 Validní dotaz na NTP server. 39
2.9 Validní odpověď NTP serveru. 39
2.10 Validní dotaz klienta. 40
2.11 Podvržená odpověď IT/OT uzlu. 40
2.12 Zaznamenání GOOSE paketů. 42
2.13 Přehrání GOOSE paketů. 43
2.14 Validní GOOSE paket stanice A. 43
2.15 Paket útoku GOOSE replay. 44
2.16 Paket nesoucí 100 položek v datové části. 46
2.17 Rychlost provozu generovaných paketů se 100 datovými položkami. . 47
2.18 Paket nesoucí pouze jednu položku v datové části. 48
2.19 Rychlost generovaného provozu s malým množstvím dat. 48
2.20 Vytížení procesoru koncentrátoru. 48
2.21 Rychlost vstupu s malým množstvím dat do koncentrátoru. 49
2.22 Objem generovaných dat při spuštění dvou instancí útoku. 49
2.23 Objem koncentrátorem přijímaných dat při spuštění dvou instancí útoku. 50
2.24 Objem generovaných dat při spuštění tří instancí útoku. 50
2.25 Záznam síťového provozu během útoku GOOSE stNum. 53
2.26 Validní pakety obsahující data. 53
2.27 Podvržené pakety bez dat. 53
2.28 Srovnání velikostí validního a podvrženého paketu. 56
2.29 Časové razítko, stNum a datová část validního paketu. 57
2.30 Časové razítko, stNum a datová část podvrženého paketu. 58
2.31 Topologie modelové sítě. 59

2.32 Heslo v textovém formátu. 60
2.33 Spuštění klienta s IP adresou MMS serveru a heslem. 60
2.34 Zachycení spojení pravým MMS serverem. 61
2.35 Spuštění klienta s IP adresou MMS serveru a heslem po provedení útoku ARP poison. 61
2.36 Zachycení spojení pravým MMS serverem i přes otrávení ARP záznamů. 61
2.37 Tabulka pravidel firewallu. 62
2.38 Spuštění klienta po úspěšném MITM útoku s heslem "Password capture". 62
2.39 Komunikace přesměrována na útočníkovu stanici zachytávající hesla. . 63

Seznam tabulek

2.1 IP adresy testovací infrastruktury. 34
2.2 Konstanty identifikující typ položky protokolu GOOSE. 67
2.3 Srovnání výkonu aplikace pro generování HMAC s a bez režimu pro maskování spotřeby. 72

Seznam výpisů

2.1 Přidané funkce do knihovny libiec61850. 36
2.2 Přijetí podvrženého paketu koncentrátorem. 45
2.3 Zamítnutí validního paketu. 45
2.4 Výstup koncentrátoru po zahlcení vstupních bufferů. 47
2.5 Přijetí podvrženého paketu od útočníka s čítačem `stNum`. 51
2.6 Zamítavá reakce koncentrátoru po přijetí validního `stNum`. 52
2.7 Reakce koncentrátoru na podvržený paket 55
2.8 Reakce koncentrátoru na validní paket po přijetí podvrženého paketu. 55
2.9 Parametry utility `ettercap` pro provedení ARP poison útoku. 61
2.10 Nastavení přeposílání komunikace mezi klientem a serverem 61
2.11 Nastavení pravidla firewallu pro přesměrování komunikace z portu 102 na port 102 útočícího stroje. 62
2.12 Příklad konfiguračního souboru. 66
2.13 Příklad volání šablony `get_item`. 66
2.14 Příklady spouštění autorem napsaného nástroje. 70

Úvod

Tématem této diplomové práce je testování zranitelností v průmyslových sítích. Po diskusi s vedoucím práce, Ing. Blažkem, byla zaměřena na energetiku, konkrétně standard IEC 61850.

Zároveň s intenzivním rozšířením internetu na konci devadesátých let dvacátého století vznikaly snahy připojit k nové technologii i kritickou infrastrukturu. Ačkoliv se jedná o úctyhodnou snahu, která má své výhody, přináší i jistá úskalí, se kterými je nutno se vyrovnat.

Výhody jsou nezanedbatelné, přináší totiž optimalizaci výrobních procesů, například v podobě jednodušší a rychlejší správy systému, rychlejším šíření informací o chybách a závadách v systému, které umožňují rychlejší zásah a tím minimalizaci ztrát. Dalšími výhodami se ukazují být možnosti vzdálené správy a rozsáhlé automatizace, která umožňuje snížení počtů obsluhujícího personálu, ačkoliv tento personál musí být vhodně vyškolen, společnost provozující dané systémy ušetří na mzdových nákladech. Zároveň čím méně lidí bude v kritické infrastruktuře pracovat, tím menší šance pro vznik závady z důvodu lidské chyby. Z provedených průzkumů zabývajících se kybernetickou bezpečností vyplynulo, že lidský faktor hraje významnou roli v počátcích útoků. V neposlední řadě se jedná také i o jednodušší a rychlejší aktualizace jednotlivých částí systému, které jsou důležité pro udržení kroku s technologickým pokrokem.

Avšak jako každá mince, i tato má dvě strany. Druhou stranou je možná bezpečnostní kompromitace třetí stranou, která, narozdíl od osobních zařízení, má dalekosáhlé důsledky. Když dojde ke kompromitaci soukromých zařízení fyzických osob, jedná se o relativně malý, rozsahem omezený, problém, který se dá celkem jednoduše řešit a pravděpodobnost katastrofálních následků pro společnost je minimální. Bohužel stejně se nemůžeme vyjádřit o útocích na kritickou infrastrukturu, do které patří i energetika, která v případě poškození může zasáhnout celé státy. Ačkoliv jsou jednotlivé prvky systémů obsažených kritické infrastruktuře navrhovány s ohledem nejen na riziko výpadku či poškození, stále může dojít rozsáhlým škodám na majetku, to v lepším případě, anebo ke ztrátám na životech v tom horším. Zmíněné útoky se již staly, a opravdu docházelo ke ztrátám na životech. Jako příklad může sloužit tzv. severovýchodní blackout v roce 2003, kdy zemřelo téměř 100 lidí.

Více o těchto událostech je uvedeno v teoretické části, kde je čtenáři přiblížena problematika fungování kritické infrastruktury a její zabezpečení proti útokům. V praktické části je čtenář seznámen s implementací vybraných útoků, pro které jsou následně navržena mitigační opatření.

1 Teoretická část studentské práce

V teoretické části se zaměříme na vysvětlení pojmů, se kterými budeme pracovat. Popíšeme si důležitost kritické infrastruktury, jak do ní zapadá energetika a přenosová soustava, následky jejího poškození včetně některých příkladů z historie. Následně si vysvětlíme, co je to ICS Cyber Kill Chain a jaký má význam, včetně ukázky mapování útoku. Nakonec si přiblížíme některé konkrétní škodlivé programy, včetně jejich využití v konkrétních útocích.

1.1 Kritická infrastruktura

Kritická infrastruktura je definována v zákonech České republiky. Na webových stránkách Ministerstva Vnitra České republiky je definována následovně:

Kritickou infrastrukturou (KI) se dle zákona č. 240/2000 Sb., o krizovém řízení a o změně některých zákonů (krizový zákon) rozumí prvek kritické infrastruktury nebo systém prvků kritické infrastruktury, jehož narušení by mělo závažný dopad na bezpečnost státu, zabezpečení základních životních potřeb obyvatelstva, zdraví osob nebo ekonomiku státu. Provozovatelem prvků KI jsou státní instituce nebo soukromé subjekty [1].

Prvkem KI je zejména stavba, zařízení, prostředek nebo veřejná infrastruktura, určené podle průřezových a odvětvových kritérií. Ministerstvo vnitra (Generální ředitelství Hasičského záchranného sboru ČR) vede seznam prvků kritické infrastruktury. V současné době je těchto prvků cca 1300 [1].

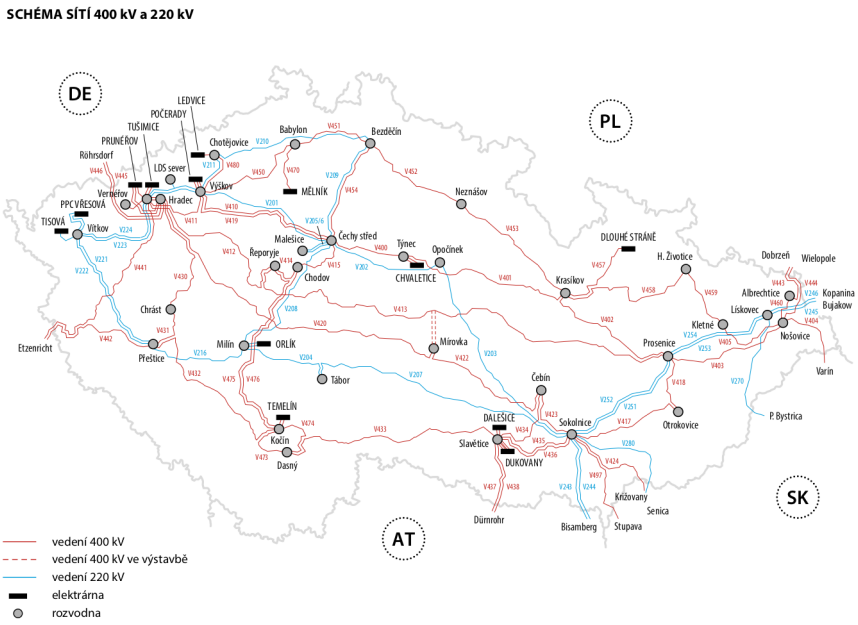
Aby mohl být určen prvek KI, musí splňovat průřezová a odvětvová kritéria určená nařízením vlády č. 432/2010 Sb. o kritériích pro určení prvku kritické infrastruktury. Průřezovým kritériem je hledisko obětí s mezní hodnotou více než 250 mrtvých nebo více než 2500 osob s následnou hospitalizací po dobu delší než 24 hodin, ekonomického dopadu s mezní hodnotou hospodářské ztráty státu vyšší než 0,5% hrubého domácího produktu nebo dopadu na veřejnost s mezní hodnotou rozsáhlého omezení poskytování nezbytných služeb nebo jiného závažného zásahu do každodenního života postihujícího více než 125 000 osob. Mezi odvětvová kritéria patří například energetika, vodní hospodářství, potravinářství a zemědělství, doprava nebo veřejná správa. Do veřejné správy pak patří například sociální zabezpečení, státní sociální podpora nebo sociální pomoc [1].

1.1.1 Energetická kritická infrastruktura

Naše civilizace je závislá na elektrické energii. Jelikož jsou dodávky elektrické energie nutnou podmínkou pro správnou funkci veškeré navazující infrastruktury, musí

být zajištěna její odolnost vůči veškerým vlivům, které by ovlivnily její správnou funkcionalitu. Jedná se např. o ochranu proti přírodním podmínkám, proti selhání komponent, proti fyzickému poškození a proti kybernetickým hrozbám. Nás zajímá poslední jmenovaná hrozba.

Přenosová soustava České republiky Na obrázku 1.1 můžeme vidět přenosovou soustavu České republiky. V přenosové soustavě tohoto státu se nachází 13 elektráren o celkovém instalovaném výkonu 22 264 MW [2]. Přenosová soustava dále obsahuje dvacet osm 400 kV, čtrnáct 220 kV rozvoden a jednu 110 kV rozvodnu [2]. V síti je 3735 km 400 kV, 1909 km 220 kV a 84 km 110 kV vedení. Hraniční přechody jsou chráněny čtyřmi 400 kV PS (Phase Shifting) transformátory. PS transformátory regulují průtok elektrického proudu z a do naší přenosové soustavy. Jedná se o ochranné opatření proti vysokým neplánovaným tokům mezi severním Německem a jeho sousedními státy, které bylo zavedeno z důvodu absence fyzického propojení v Německé přenosové soustavě.



Obr. 1.1: Rozvodná síť České republiky, převzato z [2].

1.1.2 Následky poškození kritické infrastruktury

Následky poškození kritické infrastruktury jsou nebezpečné jak z ekonomického, tak ze sociálního hlediska. Nejenže dochází k vysokým finančním ztrátám, ale může dojít i ke zranění lidí, dokonce úmrtím. Proto je třeba zabezpečit stálé dodávky služeb. V odstavci níže lze vidět následky softwarové chyby.

Severovýchodní blackout 2003 Typickým příkladem vážnosti následků poškození kritické infrastruktury, v tomto případě rozvodné sítě, je severovýchodní blackout [3]. K události došlo 14. srpna 2003 ačkoliv některé oblasti byly bez proudu pouhé dva dny, ke kompletnímu obnovení dodávek došlo až o dva týdny později, 28. srpna. Prvotní problém byl způsoben špatnou synchronizací v řídícím software, kde došlo k souběhu (race condition).

K výpadku došlo krátce po šestnácté hodině 14. srpna. Na kanadské straně byla zasažena provincie Ontario a na straně Spojených států Amerických byly zasaženy státy New York, New Jersey, Maryland, Connecticut, Massachusetts, Michigan, Ohio a Pennsylvania. Postižené státy můžeme názorně vidět na obrázku 1.2.

Celkem bylo zasaženo 55 miliónů lidí. 10 miliónů v Kanadě a 45 miliónů v USA. Během incidentů bylo oznámeno 3000 požárů, hlavně kvůli neodborné manipulace se svíčkami. Záchranné složky odpověděly na 80 000 volání o pomoc, což je dvojnásobné množství oproti standardní situaci. Bohužel během incidentu zemřelo téměř 100 lidí.

Během výpadku došlo k zasažení energetické infrastruktury, kdy bylo nutno utlumit provoz jaderných elektráren do doby, než bylo bezpečné je opatrně vrátit do normálního provozu. Taktéž došlo ke spuštění vodních a uhelných elektráren pro poskytnutí elektrické energie do nejbližších oblastí.

Z důvodu výpadku došlo k omezení až odstřihnutí od dodávek vody, jelikož pumpy nebyly napájeny. Ztráta tlaku ve vodovodní síti způsobila kontaminaci pitné vody a 4 miliónům lidí v Detroitu bylo doporučeno převařovat veškerou vodu až do 18. srpna. V okrese Macomb ve státě Michigan došlo k dočasnému uzavření 2300 restaurací, dokud nedojde k jejich dekontaminaci. 20 lidí tvrdilo, že onemocnělo po koupání v řece St. Clair. Až pět dní po výpadku dodávek elektrické energie se zjistilo, že došlo k úniku 140 kilogramů vinylchloridu z chemické továrny ve městě Sarnia. V Clevelandu a New Yorku se vylila kanalizace do vodních kanálů. V Newarku, New Jersey a dalších severních městech došlo ke kontaminaci řek Passaic a Hackensack, které vedou do Atlantiku.

Severně od Philadelphie došlo ke kompletnímu zastavení vlakové dopravy. Vlaky z a do New Yorku byly zastaveny, další den byly na páteřních linkách nasazeny lokomotivy fungující pouze na dieselový pohon. Kvůli výpadku bezpečnostních opatření byla uzavřena regionální letiště. Čerpací stanice musely být uzavřeny, protože nefungovaly pumpy. Ropné rafinerie na východním pobřeží musely být odstaveny a obnova produkce paliva byla zdlouhavá, což vedlo ke zvýšení cen a v Kanadě k přistoupení na přídělový systém.

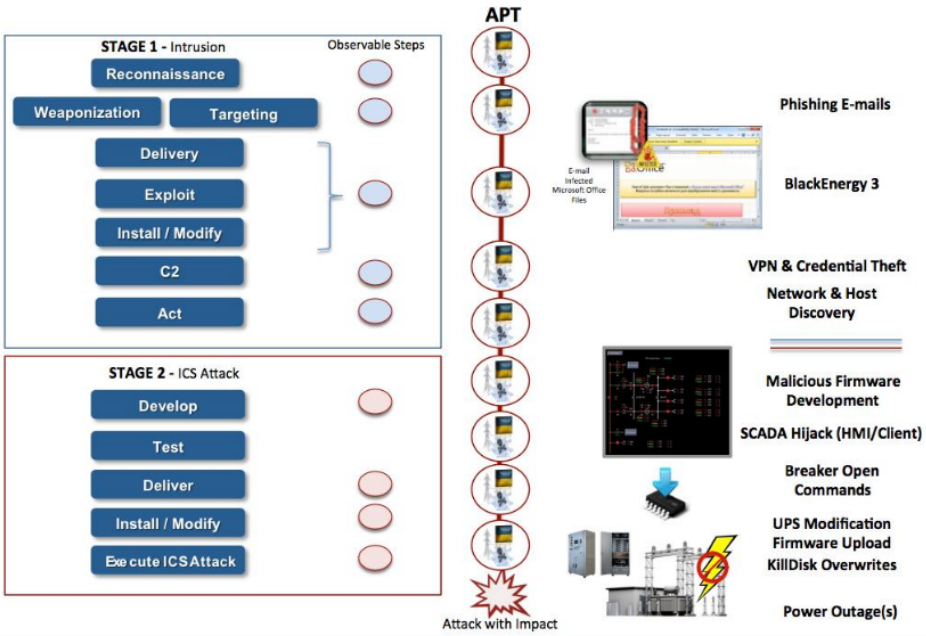
Co se týká komunikační sítě, došlo k výpadku stanic, kterým došlo palivo pro generátory. Pevné linky fungovaly dále, ale docházelo k přetížení linek. Vypadla i kabelová televize a několik internetových služeb. Informace byly dostupné přes rádiovým vysíláním. Ačkoliv výpadek elektrického proudu postihl relativně malou

plochu, výpadek internetových služeb, které běžely na zařízeních postiženém výpadkem, ovlivnil mnohem větší plochu. Výroba v továrnách musela být omezena a mnohé továrny byly zavřeny z důvodu nutnosti šetřit energií do doby, než došlo ke stabilizaci energetické sítě.

Obr. 1.2: Zasažené státy při severovýchodním blackoutu v roce 2003, převzato z [4].

1.2 ICS Kill Chain

Abychom mohli pracovat s jednotlivými útoky, musíme si stručně popsat pojem ICS Kill Chain. ICS Kill Chain, kde ICS znamená Industrial Control System [5], je několikafázový postup útoků na průmyslová zařízení. Nejedná se o samostatný útok, ale o sérii koordinovaných útoků rozdělených do fází, kde každá má svůj význam. Kompletní rozdělení lze vidět na obrázku 1.3. Jak můžeme vidět, fáze 1 se dá klasifikovat jako průmyslová špionáž případně výzvědná operace, tato fáze má několik kroků, které si popíšeme níže.



Obr. 1.3: Rozdělení ICS KC a mapování útoku na Ivano-Frankivsk v roce 2015, převzato z [5].

1.2.1 Fáze 1

Fáze 1 (Stage 1) [5] slouží výlučně k prozkoumání a naplánování útoku včetně průniku do hostitelské sítě kvůli vytvoření, použijeme-li vojenskou terminologii, předmostí, z kterého následně vykonáme veškeré zamýšlené akce.

Plánování (Reconnaissance) Během této fáze útočník shromažďuje informace o cíli svého útoku, mnohdy se jedná o využití informací z veřejně přístupných zdrojů, jako jsou zdroje indexovány službou Google search, případně Shodan. Nezřídka se jedná i o sledování sociálních sítí a veřejných dokumentů dané společnosti. Cílem tohoto kroku je odhalení možných zranitelností, na které by se mohl útočník zaměřit, a které by mohl zneužít k průniku do komponent systému. V rámci pasivního

sledování se útočník zaměřuje např. na veřejně přístupné body systému, kde hledá vhodné vzorce pro napadení a určení operačních systémů v daném systému.

Příprava (Weaponisation and Targeting) V rámci přípravy se bavíme o pojmech, v angličtině uváděným jako "Weaponisation"a "Targeting". Toto obnáší modifikaci normálně neškodných souborů, například soubory aplikace MS Office, k získání přístupu do cílového systému např. pomocí maker. Zacílení (Targeting) je proces analýzy, určení priorit cílů, spojování vhodných škodlivých a neškodných akcí pro způsobení žádaného efektu. Jako časté metriky se používá zhodnocení časové náročnosti a námahy v kombinaci s šancí na úspěšné provedení útoku, aniž by byl útočník detekován. K zneužití infikovaných souborů nemusí vůbec dojít, jelikož útočník v rámci zacílení získal přístup např. k VPN (Virtual Private Network) společnosti pomocí phishingu[1].

Průnik (Delivery, Exploit, Install/Modify) Aby mohl útočník proniknout do sítě oběti, musí mu nějakým způsobem doručit infikovaný soubor. Standardní postup používaný v této situaci je phishing. Následuje samotný exploit[2], pomocí kterého získá útočník přístup k systému, kde si nainstaluje zadní vrátka.

C2 C2, neboli Command and Control, se anglicky označuje krok, ve kterém útočník pomocí dříve nainstalovaných zadních vrátek převezme perzistentní kontrolu nad systémem. Zároveň vytvoří několik komunikačních kanálu, aby ztížil své odhalení správcem systému pomocí analýzy příchozí a odchozí komunikace.

Průzkum sítě (Act) V poslední fázi, kterou můžeme na obr 1.3 vidět pojmenovanou "Act", se útočník připravuje na přechod do druhé části samotného útoku. Jedná se o skenování sítě, zkoumání její topologie, hledání zranitelností v jednotlivých zařízeních, posílení svého maskování před správci sítě atd. Vypisovat jednotlivé činnosti by bylo zbytečně zdlouhavé, jelikož právě v této fázi se odehrává veškerá příprava na ostrou část útoku, je proto kritické, aby nebyla podceněna a byla dotažena do konce.

1.2.2 Fáze 2

Vývoj a ladění útoku (Develop) Během tohoto kroku dojde k vývoji útoku na míru danému hostitelskému systému, kdy útočník vyvine schopnosti útočit na specifika konkrétní implementace za získáním kýženého efektu. Útočník tak činí na základě uniklých dat, která získal v rámci Stage 1.

[1] Pokus o donucení uživatele zadat přihlašovací údaje do útočníkovy aplikace.

[2] Zneužití chyby.

Validace (Test) Kvůli testování zvolených vektorů útoku je prodleva mezi Stage 1 a Stage 2 vysoce proměnlivá a řádově se pohybuje v měsících. V tomto kroku musí útočník otestovat svou schopnost útočit na systém, aby docílil kýženého efektu. Je zbytečné a z pohledu útočníka nežádoucí, aby se odkryl před samotným útokem. Proto tato validace nejčastěji probíhá v síti útočníka, kde si může defacto dělat co chce, aniž by riskoval odhalení.

Doručení malware (Deliver) Po vývoji a otestování vektorů útoku musí útočník dostat svůj SW do hostitelské sítě, jinak mu je k ničemu. K tomu právě využije perzistentních komunikačních kanálů, které vytvořil v předchozích krocích.

Instalace (Install/Modify) Nyní útočník přistoupí k instalaci svého modifikovaného programového vybavení do klíčového hardware, na který chce útočit.

Ostré provedení útoku (Execute ICS Attack) Nyní může útočník přejít k samotnému provedení útoku. V momentě jeho spuštění je z pohledu útočníka mít připravené podpůrné útoky k oddálení schopnosti reakce ze strany hostitele. Ukázkový příklad těchto podpůrných útoků je uveden v sekci o útoku na Ukrajinskou energetickou síť v části 1.4.2, může se jednat například o podpůrné DoS (Denial of Service)[3] útoky, případně jiné způsoby ztěžující komunikaci. Vše závisí na dané situaci a kreativitě útočníka.

1.3 Známý malware

V této sekci se podíváme na některé známé malware, které se v posledních dvaceti letech objevily. Vzhledem k tomu, že průmyslové sítě fungují na starých legacy[4] systémech, které nejsou aktualizovány, a tedy obsahují spoustu zranitelností. Mnohdy běží průmyslové systémy na operačních systémech Windows XP, kterým skončila veškerá podpora v roce 2014, a ani nejsou k řečenému datu aktualizovány. V kombinaci s chybami, které dělají lidé, mohou být následky fatální.

1.3.1 BlackEnergy

Black Energy je rodina malware ruského původu, jehož první verze byla poprvé spatřena v roce 2007. Jednalo se o relativně jednoduchý HTTP DDoS trojského koně [6]. Využíval botnety v Rusku a Malajsii, avšak nejvíce útoku s jeho pomocí bylo provedeno v Rusku [7]. Od svého vzniku se vyvinul v sofistikovaný malware.

[3] Útok, kdy se útočník snaží znepřístupnit internetovou službu.
[4] Stará již neaktualizovaná, avšak používaná technologie.

První rozsáhlý útok, kde byl BlackEnergy využit se odehrál v roce 2008, kdy ruští hackeři napadli 54 webových stránek komunikačních, finančních společností i stránky Gruzínské vlády, pouhé týdny před Rusko-Gruzínskou válkou. Jednalo se o první koordinovaný kybernetický útok s dalšími bojovými operacemi.

V roce 2010 došlo ke kompletnímu přepsání viru, který dostal modulární architekturu a vznikl BlackEnergy2 [8]. Přibyly moduly pro spamy, podvody a cílené útoky, zatímco původní schopnost DDoS zůstala. Flexibilní struktura malware využívá pluginů s různými schopnostmi. Pluginy mohou být staženy z řídících serverů botnetu a uloženy v zašifrovaném formátu jako ovladače na discích infikovaných počítačů. Pluginy zahrnují trojského koně, který může zničit filesystém v případě, že dostane správný příkaz, stejně jako pluginy pro odesílání spamu, získávání hesel, keyloggery atp. Bot může dále stahovat a spouštět vzdálené i lokální soubory, aktualizovat se z řídících serverů a na příkaz se smazat. Nejnebezpečnější část tohoto viru je jeho schopnost se jednoduše aktualizovat, jelikož při odhalení antivirem může útočník přepsat odhalený modul, který se poté distribuuje mezi další boty.

Na rozdíl od svého předchůdce využívá rootkit/proces-injection[5] techniky a silné šifrování. Rootkit BlackEnergy2 sdílí podobné vlastnosti jako virus Rustock.E, proto jej občas antiviry špatně označí [9]. Prvotní infikování proběhne pomocí tzv. "dropperu", který dešifruje a rozbalí binární soubor rootkitu, který nainstaluje jako službu s náhodným jménem. Data jsou komprimována algoritmem LZ77 a šifrována RC4 šifrou. Dropper[6] také obsahuje exploit pro zranitelnost zveřejněnou ve věstníku Microsoftu MS08-025 [10], kdy v případě, že uživatel má omezená práva, dojde k jejich eskalaci, aby software mohl nainstalovat ovladač pro rootkit. Rootkit má na starosti tři primární funkce [9]:

- Skrývání objektů na disku, v registrech a v paměti skrz API (Aplication Programming Interface) hooking.
- Poskytnutí metod pro moduly.
- Injekce hlavního `.dll` souboru do procesu `svchost.exe`.

Hlavní síla malware je v `.dll` souboru, který je vložen do uživatelského procesu `svchost.exe`. Tím virus vystaví své API a každý kdo jej zná pro něj může napsat své moduly a rozšířit působnost kódu. Bez modulů je funkcionalita BlackEnergy2 značně omezena.

Poslední verze BlackEnergy, verze 3, se objevila v roce 2014. Jeho změny byly menšího rázu, jednalo se hlavně o zjednodušení malwaru. Došlo ke změně instalačního procesu, kde BlackEnergy3 nepoužívá ovladače, jako používaly jeho předchozí verze, ale instalační program umístí hlavní `.dll` soubor přímo do složky s daty lokálních aplikací. Také komunikační protokol mezi jednotlivými pluginy dostal změn.

[5] Dojde k infikování procesu uživatele, pomocí kterého eskaluje práva.

[6] Druh trojského koně instalující dodatečný malware.

Má převážně stejné schopnosti jako BlackEnergy2. K jeho nejvýznamnějšímu použití došlo během útoku na Ukrajinskou rozvodnou síť v oblasti Ivano-Frankivsk. Incident je blíže popsán v sekci 1.4.2.

1.3.2 Industroyer

Industroyer je sofistikovaný software navržený pro narušení výrobních procesů v průmyslových řídících systémech, přesněji v elektrických rozvodnách. Autoři implementovali podporu čtyřech průmyslových protokolů [11]:

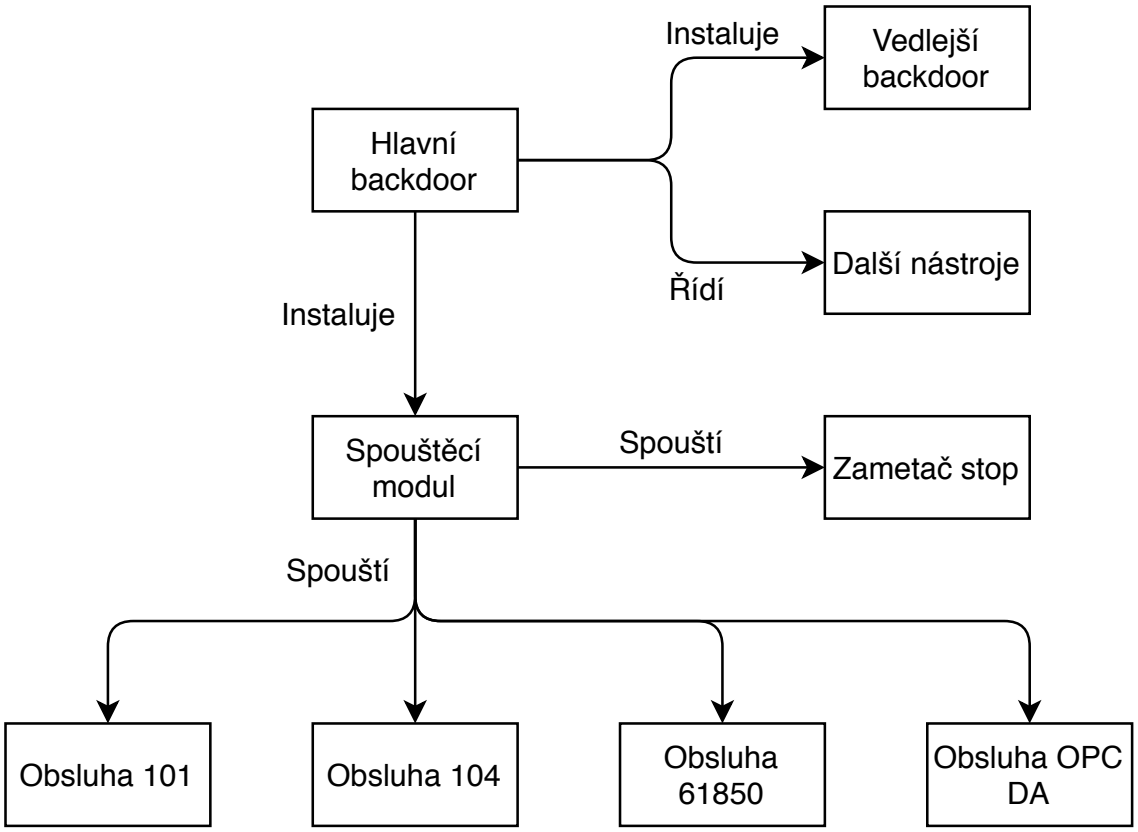
- IEC 60870-5-101 (dále 101),
- IEC 60870-5-104 (dále 104),
- IEC 61850,
- OLE pro Process Control Data Access (OPC DA).

Malware dále obsahuje nástroj umožňující DoS útoky proti některým ochranným mechanismům, hlavně proti Siemens SIPROTEC. V porovnání se softwarem použitým v útocích na Ukrajinskou přenosovou soustavu, viz sekce 1.4.2, je Industroyer výrazně pokročilejší, jelikož umožňuje ovládat přímo přepínače a jističe.

Jak můžeme vidět na obrázku 1.4, jádrem Industroyeru jsou zadní vrátka umožňující ovládat další komponenty malware. Jakožto backdoor se chová celkem přímočaře, kdy se pomocí HTTP připojí k řídícímu serveru, z kterého dostává příkazy od útočníků. Když dojde ke spojení s řídícím serverem, odešle program skrz `POST` žádost následující data: GUID (Globally Unique Identifier) řetězec pro současný hardwarový profil, verzi malware, ID vzorku a výsledek předchozích příkazů. Hlavní backdoor se po infiltraci do systému pokusí eskalovat práva, a když se mu to povede, povýší nainstalovanou verzi na službu systému Windows pomocí nahrazení hodnoty registru `ImagePath` nekritické služby cestou k novému spustitelnému souboru. Program dále nainstaluje druhé, záložní, zadní vrátka pro případ, že budou ta hlavní odhalena a uzavřena. Jedná se o infikování aplikace Poznámkový blok. V momentě, kdy získá malware administrátorská práva, dojde k nahrazení původního Poznámkového bloku jeho plně funkční infikovanou verzí. V momentě spuštění tohoto programu dojde k vložení kódu pro shell přímo do paměti a následně je vykonán. Tím dojde ke stažení potřebných komponent programu.

Soběstačný spustitelný soubor je tzv. "Launcher", který je zodpovědný za spouštění komponenty pro zametání stop a implementací pro jednotlivé protokoly.

Díky implementaci protokolu 101 mohou útočníci ovládnout RTU (Remote Terminal Unit) skrz sériovou linku. Pro každou adresu zařízení vygeneruje dva pakety a pošle je RTU. Nahrání dat modulu 101 má tři fáze. V první se pokouší zařízení vypnout, v druhé se snaží zařízení zapnout a v poslední fázi jej vypne.



Obr. 1.4: Zjednodušené schéma Industroyeru.

Komponenta pro protokol 104 umožňuje úpravu na míru pro libovolnou infrastrukturu. Po spuštění se pokusí přečíst konfigurační soubor a vytvoří vlákno pro každou stanici, která je v něm definována. Poté komponenta ukončí legitimní proces komunikující s danými stanicemi a spustí svůj vlastní, díky kterému převezme kontrolu.

Komponenta protokolu 61850 je samostatný nástroj, který má svůj spustitelný soubor a svou knihovnu DLL (Dynamic-Link Library). Po spuštění dojde k přečtení konfiguračního souboru, případně identifikaci masky sítě a pokusí se připojit na port 102 ve všech validních IP v dané síti. V případě vhodné odpovědi komponenta komunikuje pomocí MMS (Manufacturing Message Specification) zpráv.

Modul řídící OPC DA komunikaci implementuje klientskou část standardu. Nevyžaduje žádný konfigurační soubor, jelikož najde veškeré OPC servery a prohledá dílčí položky, které jsou na nich uloženy. Do těch posléze zapisuje.

Modul pro mazání stop je destruktivní modul, který je použit v poslední části útoku. Když je spuštěn, tak se pokusí vyjmenovat všechny klíče registrů v seznamu služeb `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services` [11] a přepsat jejich hodnoty `ImagePath` prázdným řetězcem, čímž znemožní spuštění operačního systému na stanici. Dalším krokem je odstranění souborů s definovanou příponou na všech interních i externích discích. Dojde k jejich přepsání náhodným obsahem paměti, pro jistotu dvakrát, aby nemohlo dojít k obnovení souborů. Pro urychlení se nepřepisují celé soubory, ale pouze jejich začátek. Následně se pokusí ukončit všechny procesy krom sama sebe, čímž vypne systém.

1.3.3 Stuxnet

V následující sekci si přiblížíme malware Stuxnet, princip jeho fungování a některé scénáře útoků. Jelikož se jedná o důsledně zdokumentovaný malware, shrnutí bude spíše v obecné rovině. Detaily lze nalézt v literatuře [12], [13] a [14].

Stuxnet byl identifikován v roce 2010 poté, co se náhodou rozšířil mimo svůj cíl, továrnu v Natanz, kvůli chybě v aktualizaci. Klonuje se pomocí přenosných disků využívaje zranitelnosti systému Windows umožňující automatické spuštění souborů na přenosném médiu. Šíří se v lokálních sítích skrze zranitelnost služby Windows Print Spooler, také skrz Sambu atd. Infikování probíhá v několika krocích.

1. Pomocí několika 0-day exploitů[7] v systémech Windows a sítích se šíří jak do hloubky (systému), tak do šířky (infikuje co nejvíce zařízení).
2. Vyhledá zařízení Step7 výrobce Siemens, do kterého pronikne, opět s pomocí využití chyb systému Windows.

[7] Zneužití obecně neznámé zranitelnosti vůči které zatím neexistuje obrana.

3. Dojde k napadení samotných PLC (Programmable Logic Controller), čímž defacto převezmou kontrolu nad celou sítí.

Pro zamaskování své přítomnosti Stuxnet úspěšně kompromitoval dva certifikáty pro digitální podpis a vypadal v očích správců systému zcela validně. Zároveň, ačkoliv měli útočníci přístup k viru pomocí řídícího serveru, klíčový počítač se Step7 programovým vybavením byl pravděpodobně odpojen od internetu, takže veškeré vlastnosti umožňující napadení a ovládnutí systému musely být zakomponovány v jeho kódu. Veškeré aktualizace probíhaly pomocí vytvoření peer-to-peer sítě uvnitř továrny samotným Stuxnetem. Nejdříve nainstaluje RPC (Remote Procedure Call) server a klienta. Dále dojde ke spuštění serveru, kterého se připojující klienti mohou zeptat, jaká verze viru na stanici běží. V případě, že na serveru běží aktuálnější verze, klient zadá požadavek o aktualizaci, virus na serveru vytvoří svou kopii a odešle ji klientovi, který se následně aktualizuje. Díky tomuto principu může dojít k infikování počítačů, které nemají odchozí internetové připojení. Co se týká napadení PLC, je vskutku vybíravý, napadá pouze PLC s proměnlivou frekvencí od výrobců Vacon (Finsko) a Fararo Paya (Irán). Dále napadá systémy, jejichž motory rotují s frekvencí mezi 807 Hz a 1210 Hz. Poté nainstaluje malware do paměti bloku DB890 PLC, které sleduje sběrnici Profibus[8], a řídí frekvenci otáčení motorů.

Nejvýznamnějším použitím tohoto viru je závod na obohacování uranu v Íránském Natanz v roce 2010, kde pomocí modifikace rychlosti otáčení motoru byli útočníci schopni způsobit zvýšené namáhání citlivého materiálu centrifug, čímž docílili jejich destrukce a zpomalením postupu v Íránském jaderném programu.

1.4 Známé útoky

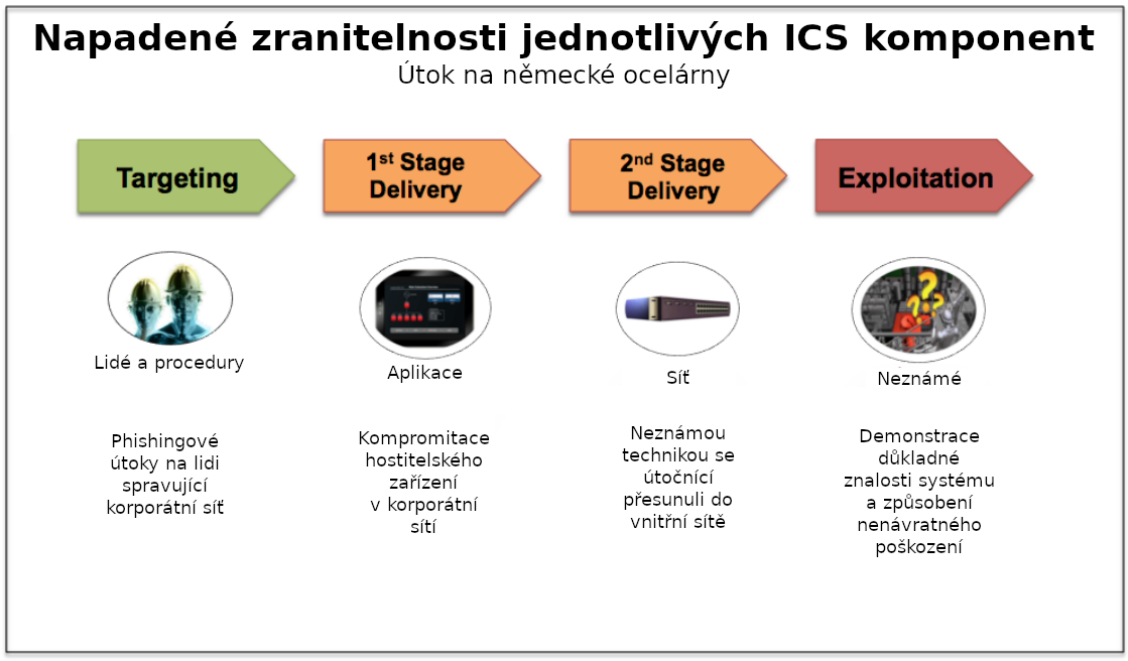
V následující sekci si popíšeme některé známe útoky na průmyslová zařízení. Nejdříve si vysvětlíme průběh útoku na německé ocelárny a následně si popíšeme útok na ukrajinskou přenosovou soustavu v roce 2015, kde jdou krásně vidět možné následky útoků na kritickou infrastrukturu.

1.4.1 Útok na německé ocelárny v roce 2014

V prosinci 2014 vypustil Federální úřad pro informační bezpečnost výroční zprávu, ve které oznámili úspěšné napadení oceláren [15]. Do tohoto útoku byl jediný zveřejněný případ kybernetického útoku, který způsobil fyzické škody, Stuxnet, který je popsán v sekci 1.3.3.

[8]Proprietární sběrnice firmy Siemens.

Prvotní útok byl veden phishingovými emaily, kdy se útočníci zaměřili na korporátní síť, konkrétně na operátory zařízení. Podobný styl byl pozorován i u útoků HAVEX a BlackEnergy2. Došlo ke kompromitaci pomocí infikovaných souborů. V momentě, kdy byly soubory otevřeny, došlo k napadení aplikace a vytvoření přístupového bodu do vnitřní sítě cíle. Následovala fáze průzkumu, kdy pomocí keyloggerů, skenování sítě a dalších technik získali útočníci přehled o zranitelnostech k provedení útoku. Kompletní průběh lze vidět na obrázku 1.5. Nejdříve došlo k infikování narušení korporátní sítě pomocí phishingových útoků, kde byl nainstalován malware na kompromitovaný stroj. Následně se útočníci přesunuli neznámou technikou do průmyslové sítě dané ocelárny, z které následně provedli útok, který nenávratně poškodil výrobní zařízení.



Obr. 1.5: Útok na německé ocelárny v roce 2014, převzato z [15].

Mezi napadenými částmi byly řídící systémy, a dokonce i vysoké pece, které nemohly být správně vypnuty a došlo k fyzickému poškození. Pravděpodobně byly zasaženy i řídící PLC, výstražné systémy, HMI (Human-Machine Interface) a další. Díky kvalitnímu návrhu průmyslového systému a profesionalitě obsluhy nedošlo ke ztrátám na životech ani zraněním.

1.4.2 Ivano-Frankivsk

Dne 23. prosince 2015 [16] došlo v Ivano-Frankivsku k výpadku elektrického proudu, který zasáhl zhruba 225 000 lidí. Výpadek nastal v 15:35 a trval do 18:35 místního času, kdy bylo vyřazeno z provozu sedm 110 kV stanic a dvacet tři 32 kV stanic.

V důsledku útoku musely stanice fungovat v manuálním režimu. Podle dostupných informací došlo k převzetí kontroly nad distribučním systémem SCADA[9] ve třech zařízeních.

Útok byl směřován na regionální distribuční síť. I před relativně rychlé obnovení dodávek energie, musely rozvodny pracovat v omezeném režimu z důvodu kontroly škod. Vyšetřováním bylo zjištěno, že útočníci plánovali tento útok dlouhodobě. K napadení stanic došlo minimálně půl roku před samotným provedením útoku, který vedl k výpadku dodávek proudu. Útočníci získali přístup díky malware BlackEnergy3, který je popsaný v samostatné sekci výše.

V první fázi útoku využili útočníci phishingu cílenému na několik společností, kde pomocí infikovaných souborů MS Office dostali do systému malware BlackEnergy3, díky kterému pak získali přihlašovací údaje do systému. Kvůli absenci dvou faktorové autentizace VPN (Virtual Private Network) bylo přihlášení do sítě triviální. Po prvním průniku pomocí nástrojů pro vzdálený přístup vytvořili útočníci několik komunikačních kanálu, mimo jiné VPN do ICS sítí, pro provádění výzvědných operací kvůli zacílení útoku. Nejsilnější stránkou útočníků bylo utajení pomocí mazání logů a využití modifikovaného KillDisku[10] pro vymazání "Master Boot Record"z vybraných stanic, přičemž se půl roku vyhýbali detekci, do útoku si jich nikdo nevšiml. Během pozorovací fáze útočníci využili neschopnost adminů neustále sledovat ICS síť a vyhledávat nepravidelnosti v provozu skrz aktivní obranná opatření.

V rámci druhé fáze útoku ovládli útočníci HMI stanice, napsali škodlivý firmware na míru pro koncová zařízení, kdy vyřadily konvertory ze sériové linky na ethernet bez možnosti opravy. Dále využili UPS (Uninterruptible Power Supply) pro zasažení provozu stanic. Během provedení ostrého útoku byl veden DDoS útok na telefonní centra společnosti, aby došlo k zamezení komunikace mezi zákazníky a společností. Došlo tedy ke zpoždění reakce na útok.

Na obrázku 1.3 lze vidět mapování útoku na ICS Kill Chain. Útok tento model přesně kopíroval. Z délky pozorování cíle se dovozuje, že cíle byly vybrány záměrně [16].

Zajímavé na tomto útoku je, že samotná zadní vrátka, KillDisky, BlackEnergy3 ani infikovaný firmware nemůže za výpadek, jelikož to byly pouze cesty k manipulaci s rozvodnou sítí. Až výsledná manipulace se sítí stála za výpadkem.

[9]Supervisory Control and Data Acquisition.

[10]Program k nenávratnému smazání disku.

2 Výsledky diplomové práce

V následující kapitole si popíšeme standardy definující protokoly, na které autor simuloval útoky. Význam daných protokolů a použití v praxi. Zároveň definujeme model infrastruktury, na které autor testoval útoky a samozřejmě vysvětlíme zvolené vektory útoků.

2.1 IEC 61850

IEC 61850 je mezinárodní standard definující protokoly pro komunikaci mezi inteligentními zařízeními ve stanicích přenosové sítě. Standard obsahuje implementaci protokolů MMS (Manufacturing Message Specification), GOOSE (Generic Object Oriented Substation Event) a SMV (Sampled Measured Values) [17]. Níže si přiblížíme obecnou strukturu definovanou ve standardu a následně si popíšeme jednotlivé protokoly, hlavně MMS a GOOSE, jelikož na tyto implementace se v praktické části zaměřujeme. Detailní specifikace je k dispozici v [17].

2.1.1 Informační model

Informační model standardu IEC61850 se skládá z několika vrstev. Vrstvy jsou reprezentovány, od shora dolů, zařízeními fyzickými, logickými, následně logickými uzly a datovými objekty. Názorná ukázka je na obrázku 2.1.

Fyzické zařízení (Physical Device) Model začíná fyzickým zařízením, někdy nazývaným IED (Inteligent Electronical Device), které je připojené do sítě. Je definováno svou síťovou adresou.

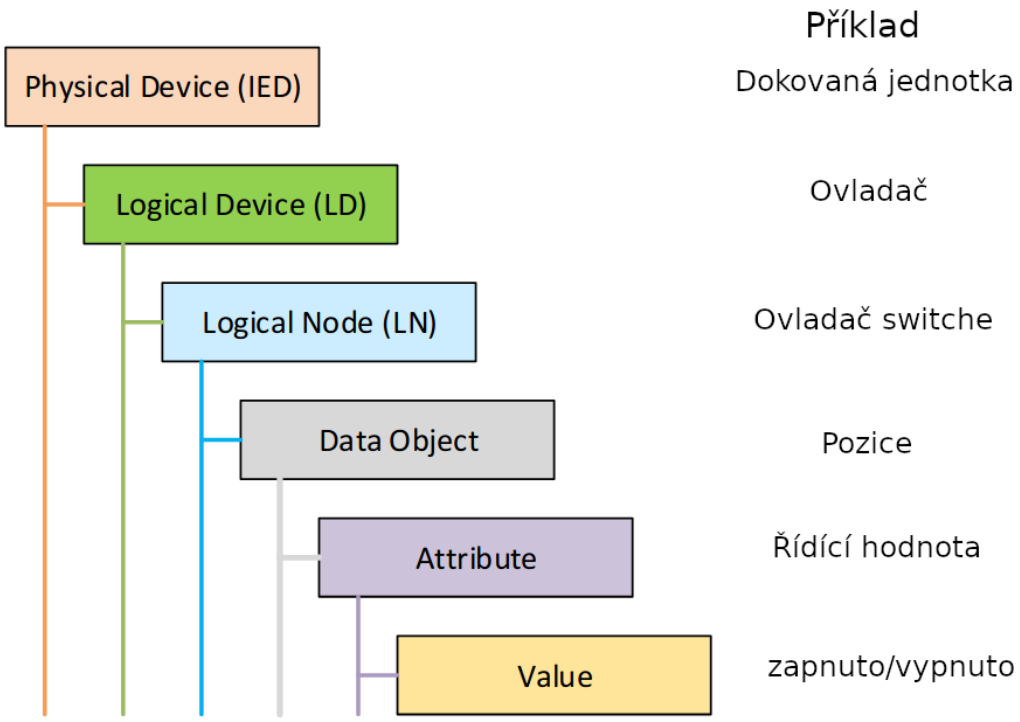
Logické zařízení (Logical Device) V každém fyzickém přístroji může být jedno a více logických strojů, které spojují data z více zdrojů do jednoho fyzického zařízení. Každé obsahuje:

- identifikátor `LDName`, který identifikuje logické zařízení na síti,
- identifikátor `LogicalNode`, který obsahuje seznam všech logických uzlů, které jsou součásti logického zařízení. Musí obsahovat alespoň jeden logický uzel, Logical Node Zero,
- službu `GetLogicalDeviceDirectory`, která vrací seznam referenčních objektů, aby mohl být každý logický uzel adresován klienty.

Logický uzel (Logical Node) Každá funkce, kterou má vybavení stanice, má přiřazený logický uzel. Jedná se o virtuální reprezentaci jednotlivých zařízení, které

shlukují data a služby, které přísluší každé funkci ve stanici. V každém logickém zařízení musí být alespoň tři logické uzly. Jsou to dva uzly spravující dané logické zařízení a jeden uzel vykonávající samotnou funkci.

Datový objekt (Data Object) Každý logický uzel obsahuje datové objekty, které reprezentují jeho aplikaci. Každý takový objekt má své jméno, jsou určeny standardem a jsou funkčně spojeny s účelem systému.



Obr. 2.1: Informační model standardu IEC61850, převzato z [17].

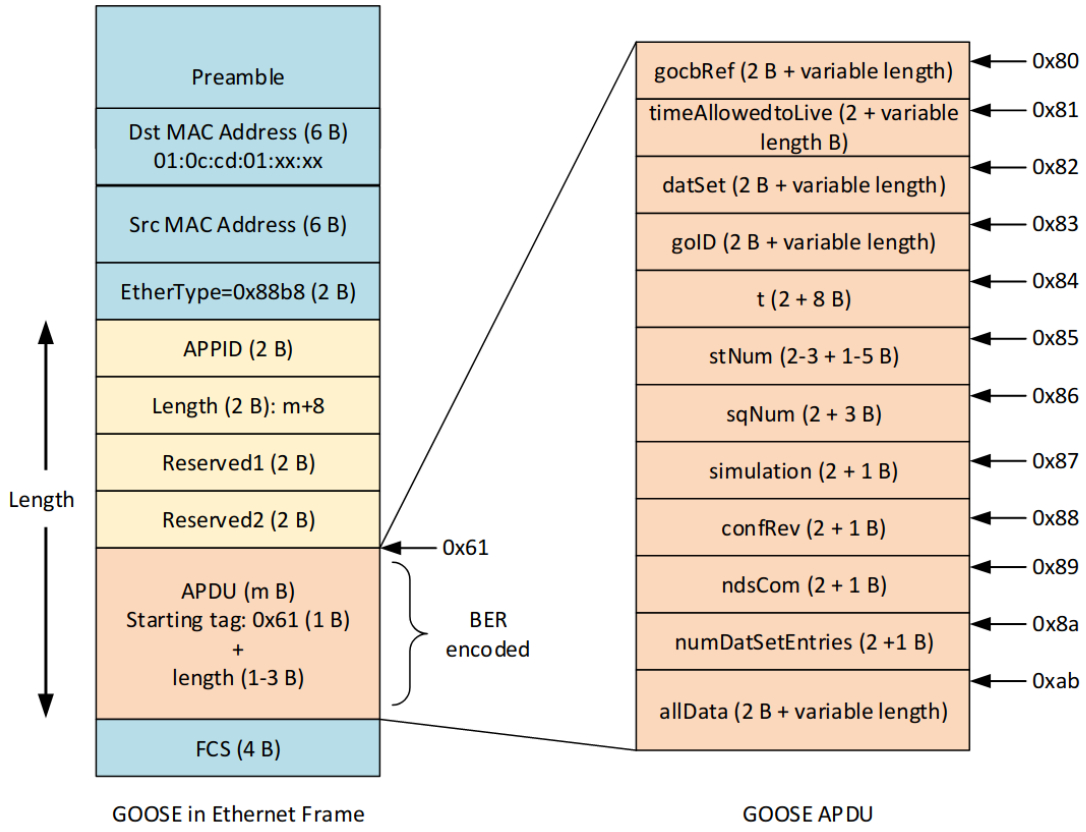
2.1.2 Protokol GOOSE

Zprávy GOOSE jsou spojeny se třemi vrstvami OSI modelu, fyzickou, síťovou a aplikační. Zapouzdření protokolu můžeme vidět na obrázku 2.2. Jak vidíme, GOOSE definuje čtyři pole pevné délky a sekvenci TLV (Type-Length-Value) dat.

Protokol se skládá z následujících položek.

- `GoCBRef` - název instance řídícího bloku ve formátu LDName/LLN0.GoCBName
- `TimeAllowedtoLive` - čas určující aktuálnost přenášených dat
- `DataSet` - pole určující data, která se budou přenášet
- `GoID` - atribut umožňující identifikaci GOOSE zprávy
- `T(imestamp)` - čas inkrementace atributu `StNum`

- `StNum` - čítač inkrementovaný pokaždé, když se změní přenášená hodnota v DataSetu
- `SqNum` - sekvenční číslo hlášení, inkrementováno po každé odeslané zprávě, po změně `StNum` dojde k vynulování
- `Simulation (test bit)` - pokud je nastaven na hodnotu 1, jedná se o simulaci
- `ConfRev` - počet revizí konfigurací formátu odesílaných dat
- `NdsCom` - určuje zda data potřebují další prověření
- `NumDatSetEntries` - počet polí v data setu
- `allData` - seznam uživatelem definovaných zpráv ve formátu `MMS NamedVariableList`

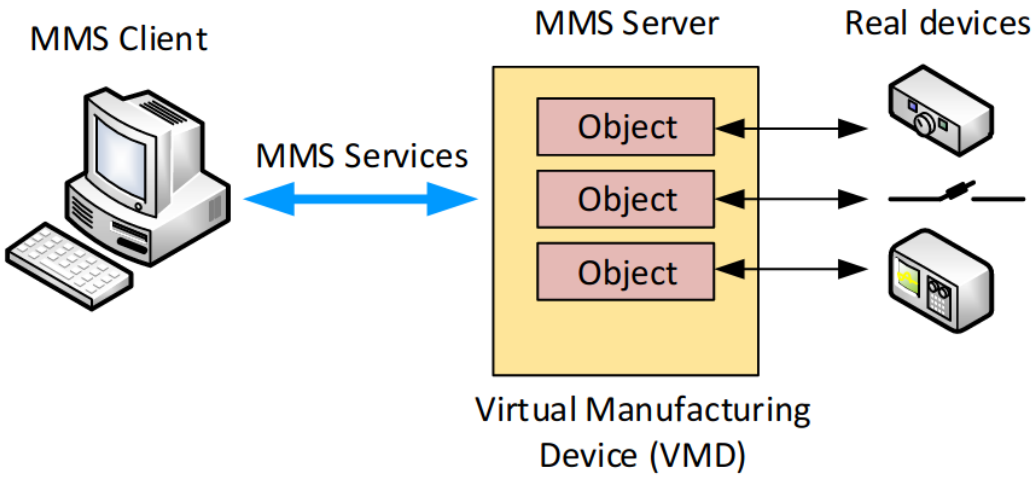


Obr. 2.2: Zapouzdření protokolu GOOSE, převzato z [17].

GOOSE zprávy slouží k rychlé výměně dat a je založena na vzoru Publish-Subscribe. Jedná se o model nepřímé komunikace, kdy odesílatelé (publishers) posílají do zvoleného komunikačního kanálu svá data, aniž by je zajímali přihlášení odběratelé. Odběratelé (subscribers) naopak z daného komunikačního kanálu odebírají zprávy, které je zajímají.

2.1.3 Protokol MMS

Manufacturing Message Specification je systém pro modelování reálných zařízení a výměnu dat mezi těmito zařízeními v reálném čase. Komunikuje pomocí modelu klient-server, kde klient je síťová aplikace, která se ptá na data a server je zařízení, které obsahuje VMD (Virtual Manufacturing Device) a dílčí objekty, ke kterým může klient přistupovat. Jak můžeme vidět na obrázku 2.3, jeden server spravuje více reálných zařízení.



Obr. 2.3: Klient-server model MMS, převzato z [17].

Pomocí zpráv MMS lze řídit chytrou síť zařízení. Tedy pokud útočník ovládne MMS server, defacto získá kontrolu nad podřízenými zařízeními.

Protokol je zapouzdřen v TCP paketech pomocí TPKT (ISO (International Organisation for Standardization) Transport Service on top of the TCP) a COTP (Connection Oriented Transport Protocol) protokolů, konkrétně na portu 102. Více informací o TPKT se nachází v RFC 1006. COTP definuje standard ISO 8073/X.244.

Samotný protokol je definován ve standardu ISO 9506. Implementuje dva typy komunikace. První typ jsou potvrzované služby. Tyto služby jsou definovány v příloze J. standardu ISO 9506. Každá žádost je potvrzená kladně, nebo záporně. Požadavek může být taky zrušen. Každý paket žádosti má své ID korelované s odpovědí. Druhým typem jsou nepotvrzované zprávy, které, jak již název napovídá, nedostávají potvrzení. Ve standardu jsou definovány tři zprávy: `InformationReport`, `unsolicitedStatus` a `eventNotification`. Tyto zprávy jsou vyvolávány serverem a nelze je zrušit.

2.2 IEC 62351

Z důvodu absence zabezpečení standardu IEC 61850 proti kybernetickým útokům se v roce 2007 začal formovat standard IEC 62351, který má tyto problémy řešit. Standard IEC 61850 se zabývá kybernetickou bezpečností řídících oblastí energetických systémů a celkovým zachováním principů důvěrnosti, integrity, dostupnosti a nepopiratelnosti, hlavně skrze zavedení autentizačních mechanismů. Standard IEC 62351 má celkem 10 částí, přičemž pro tuto diplomovou práci jsou důležité pouze dvě z nich. Konkrétně IEC 62351-4 definující bezpečnostní profily pro protokol MMS a IEC 62351-6 definující bezpečnostní opatření pro standard IEC 61850, konkrétně protokoly GOOSE a SMV [18].

Část IEC 62351-4 obsahuje doporučení pro transportní a aplikační profily. Pro aplikační profil doporučuje X.509 certifikáty, přičemž zařízení zamítne přijatou zprávu v případě, že časové razítko na přiloženém certifikátu je starší než 10 minut. Pro transportní vrstvu doporučuje použití technologie TLS (Transport Layer Security) a změny výchozího komunikačního portu. Avšak dle [18] je použití TLS pro transportní služby založeny na OSI modelu omezeno, jelikož TCP není jejich součástí, proto není transportní služba zahrnuta ve standardu. Problém s aplikačním profilem je zajištění pouze prvotní autentizace, přičemž důvěrnost a integrita nejsou pokryty, stejně jako autentizace následujících zpráv. Transportní profil navíc podporuje jak zabezpečený, tak nezabezpečený režim, který může potenciální útočník zneužít.

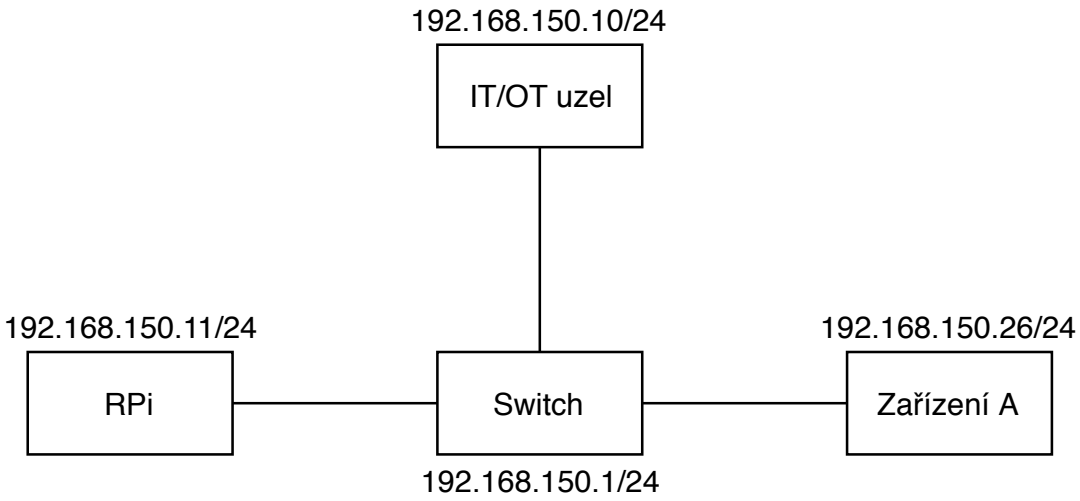
Část IEC 62351-6 obsahuje doporučení pro protokoly GOOSE a SMV. Bohužel z krátkého časového intervalu pro doručení zpráv od okamžiku události, zejména u kritických aplikací nelze aplikovat některá bezpečnostní opatření, jelikož například šifrování zpozdí doručení dat o více než 4 milisekundy, které jsou vyžadovány k doručení zpráv. V případě volnějších termínu poskytuje IEC 62351-6 rozšíření pro GOOSE a SMV zprávy, které využívají VLAN (Virtual Local Area Network), a to v podobě přidání podpisu na bázi RSA (Rivest, Shamir, Adleman) pro zaručení integrity. Bohužel žádný mechanismus pro aplikace vyžadující časy menší než 4 milisekundy standard nenabízí [18].

2.3 Model testovací infrastruktury

Pro testování zvolených zranitelností byly vytvořeny dvě testovací infrastruktury. Pro útoky na simulační bit protokolu GOOSE a NTP spoofing byla vytvořena fyzická infrastruktura v laboratoři fakulty, a pro zbývající útoky, tedy GOOSE flood, GOOSE stNum, GOOSE semantic, GOOSE replay a MMS password capture, byla vytvořena virtualizovaná infrastruktura.

2.3.1 Fyzická infrastruktura

V rámci semestrální práce byla vytvořena fyzická infrastruktura, která více než dostatečně poslouží naším účelům. Jedná se o jednoduchou síť s adresou 192.168.150.0 a maskou 255.255.255.0, která má čtyři prvky. Model můžeme vidět na obrázku 2.4. PC je útočníkem infikovaný stroj na hranici IT/OT (Informační technologie/Operační technologie) sítí, který útočník využívá jako pivot. Na PC je nainstalován Kali Linux verze `5.6.0-kali1-amd64`, distribuce sloužící k penetračnímu testování. Blok RPi reprezentuje RaspberryPi s operačním systémem Raspbian, na kterém běží NTP server synchronizující čas v celé síti. Zařízení A reprezentuje nejmenované zapůjčené zařízení běžící na operačním systému Linux, jedná se o ochranu transformátorů přenosové soustavy. Umí implementace protokolu MMS, GOOSE a SMV. Odesílaná data obsahují informace o proudu, napětí a výkonu.



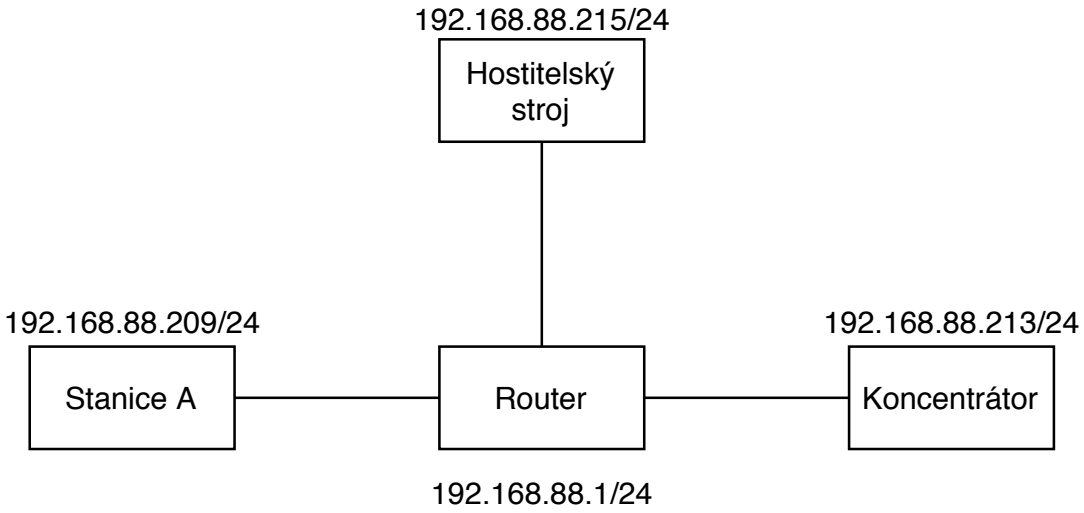
Obr. 2.4: Model fyzické infrastruktury.

2.3.2 Virtualizovaná infrastruktura

Z důvodu rozšíření koronavirové infekce, následnému vyhlášení pandemie, nařízení vlády a nucenému uzavření fakulty byla vytvořena virtualizovaná testovací infrastruktura s pomocí virtuálních strojů s operačním systémem Debian. Stroje běží ve virtuálním prostředí VMware Workstation 15 Player. Zmíněné virtuální stroje a jejich programové vybavení bylo vytvořeno v rámci bakalářské práce pana bakaláře Hudce [19]. Následně byly upraveny vedoucím této práce, Ing. Blažkem, a poskytnuty k vypracování této diplomové práce.

Infrastruktura se skládá z hostitelského počítače, na kterém běží virtualizovaný koncentrátor s IP adresou 192.168.88.213/24. Zmíněný koncentrátor přijímá zprávy

od dvou zařízení, stanice A a stanice B. Pro naše účely nám postačuje pouze virtualizovaný stroj se stanicí A s IP adresou 192.168.88.209/24. Hostitelský stroj používá operační systém Kali Linux, má adresu 192.168.88.215/24 a je připojen k routeru s adresou výchozí brány 192.168.88.1/24. Protože použitá síť obsahuje službu DHCP (Dynamic Host Configuration Protocol), může dojít ke změně IP adres. Virtualizovanou infrastrukturu můžeme vidět na blokovém schématu na obrázku 2.5 níže.



Obr. 2.5: Model virtualizované infrastruktury.

Hardwarové vybavení virtualizované infrastruktury je následující. Hostitelský stroj má šesti jádrový 64 bitový procesor AMD Ryzen 5 1600 s dvanácti virtuálními jádry, které jsou taktovány na 3.2 GHz (gigahertz), 16 GB (gigabyte) operační paměti a gigabitovou síťovou kartou. Každý virtuální stroj má přiděleno jedno jádro, 512 MB (megabyte) operační paměti a využívá gigabitovou síťovou kartu hostitelského stroje.

2.4 Útoky

V této sekci si popíšeme konkrétní útoky na testovací infrastrukturu. Nejdříve si řekneme něco o teoretických scénářích, možných vektorech útoků a následně si popíšeme implementaci. Jako první začneme útokem na simulační bit protokolu GOOSE viz 2.1.2, pokračovat budeme podvržením NTP (Network Time Protocol) serveru, kdy se autor pokusil přimět zařízení A ke změně svého vnitřního času. Následuje série útoků na protokol GOOSE, konkrétně útok GOOSE replay, GOOSE flood, GOOSE stNum, GOOSE semantic. Útoky jsou zakončeny útokem MMS password capture.

2.4.1 Útok na simulační bit protokolu GOOSE

Z důvodu zachování rychlé odezvy systému není komunikace protokolem GOOSE šifrovaná. Data jsou odesílaná na druhé vrstvě v otevřeném formátu a kdokoli si je může přečíst. Nejenže hrozí riziko úniku informací o stavu systému, ale kdokoliv může daná data zachytit, modifikovat a provést úspěšný MITM (Man in the Middle) útok. Přesně o tohle se pokusil autor této práce.

Teoretický scénář Představme si situaci, kdy dochází například ke kritickému přepětí v přenosové soustavě. V rozvodnách jsou umístěny sondy, které mají dispečery upozornit na tyto výkyvy. Najednou sondy začnou hlásit kritické hodnoty, ale jelikož je zapnutý příznak simulace, k dispečerovi se dostane informace, že se jedná pouze o testovací data. Bohužel data jsou reálná, automatické systémy zajišťující ochranu sítě nezareagují a dispečeři budou mít ztíženou možnost reakce, jistě dojde k jejímu zpoždění. Během reakční doby může dojít k nezvratnému poškození přenosové sítě se všemi následky, které jsme si ukázali v předchozích kapitolách.

Další možný scénář se odehrává v jaderné elektrárně, kde jsou použitá zařízení komunikující pomocí protokolu GOOSE ke sledování stavu jaderného reaktoru. V předchozím odstavci jsme si naznačili možnou situaci v síti, kde je na reakci relativně dost času. V případě jaderného reaktoru budou následky útoku katastrofální.

Vektor útoku V našem teoretickém scénáři útok zahájíme převzetím kontroly nad vstupním bodem IT/OT sítě. Například eskalací práv na systému linux, což nám umožní vytvořit pivot pro následný průnik do OT sítě. Konkrétní nastavení sítě `192.168.150.0/24` lze vidět v tabulce 2.1.

Tab. 2.1: IP adresy testovací infrastruktury.

Prvek sítě	IP
Switch	192.168.150.1
IT/OT uzel	192.168.150.10
Raspberry Pi	192.168.150.11
Zařízení A	192.168.150.26

Po prostudování odesílaných dat napsal autor tři programy zachytávající odesílaná data. Z bezpečnostních důvodů, po konzultaci s vedoucím Ing. Blažkem, není možné zveřejnit konkrétní MAC adresy zařízení A, ani MAC adresy, na které odesílá zachycená data.

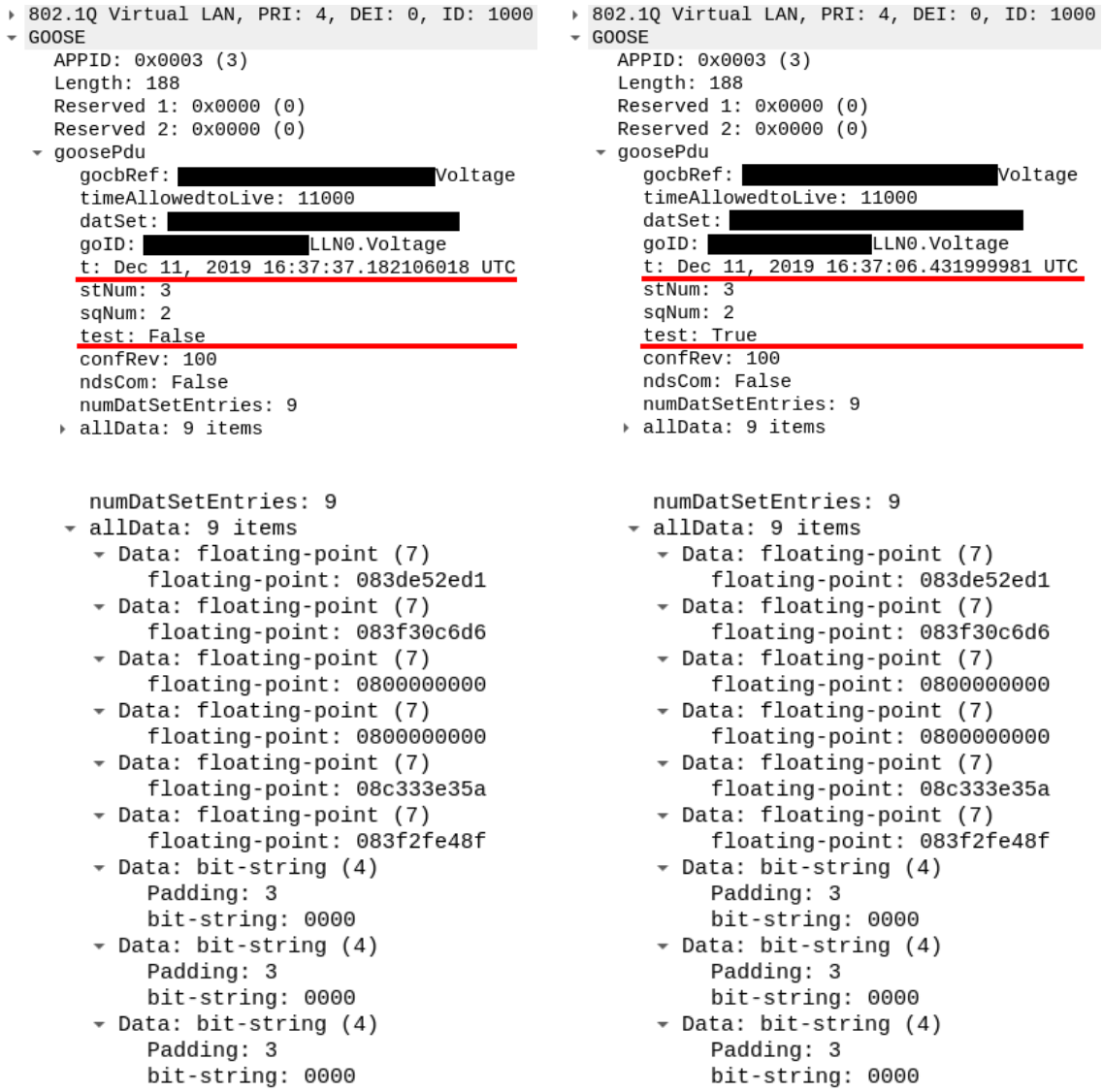
Autor využil a modifikoval knihovnu `libIEC61850`[1]. Zmíněné modifikace se týkají zpracování dat, kdy knihovna nesplňovala přesné požadavky, konkrétně se jedná o settery `StNum` a `SqNum`. Z pohledu knihovny dává smysl, aby docházelo k modifikaci těchto interních proměnných automaticky, avšak pro provedení útoku je třeba přesně zkopírovat data, která jsou v paketu obsažena. Změny jde vidět na výpisu 2.1.

Pomocí modifikované knihovny vytvořil autor program pro každý logický uzel zařízení A, jehož pakety zachytával, přečetl, modifikoval simulační bit, zabalil a znovu odeslal na původní MAC adresu. Zároveň je třeba si uvědomit, že nemůžeme jen tak odesílat data na druhé vrstvě, aniž bychom změnili MAC adresu infikovaného zařízení. Musíme proto pomocí utility změnit MAC adresu infikovaného počítače na MAC adresu fyzického zařízení. Nyní v síti máme dva pakety, oba se správnými daty, správnými časovými značkami, ale jeden obsahuje zapnutý simulační příznak. Už teď dokážeme způsobit nekonzistenci systému. Na obrázku 2.6 můžeme vidět podobu originálního paketu a paketu s modifikovaným test bitem. Z důvodu nemožnosti zveřejnit identitu zařízení A jsou MAC adresy a další identifikační údaje cenzurovány.

[1] https://github.com/mz-automation/libiec61850

Výpis 2.1: Přidané funkce do knihovny libiec61850.

```
void
GoosePublisher_setStNum(GoosePublisher self, uint32_t
   stNum)
{
self->stNum = stNum;
}

void
GoosePublisher_decStNum(GoosePublisher self)
{
self->stNum--;
}

void
GoosePublisher_setSqNum(GoosePublisher self, uint32_t
   sqNum)
{
self->sqNum = sqNum;
}

void
GoosePublisher_decSqNum(GoosePublisher self)
{
self->sqNum--;
}
```



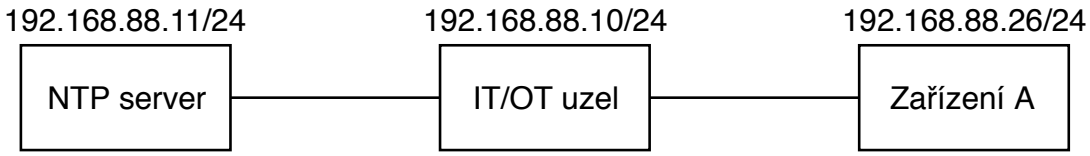
Obr. 2.6: Originální pakety (vlevo) a podvržené pakety (vpravo).

2.4.2 NTP Spoofing

Vzhledem k faktu absence šifrování komunikace pomocí NTP ji může útočník libovolně odposlouchávat a modifikovat. Opět může z důvodu časové desynchronizace vzniknout škoda ve výrobním procesu.

Teoretický scénář Mějme systém kontrolních mechanismů, který kvůli automatizaci odezvy závisí na přesné časové synchronizaci jednotlivých složek. Úspěšný NTP spoofing může jako jeden z podpůrných útoků ovlivnit reakci automatizovaných systémů a způsobit zpoždění reakce obsluhy, stejně jako ovlivnit zpracování údajů z napadených senzorů. Opět může dojít ke katastrofálním následkům, jelikož správná časová synchronizace je kritická pro běh mnoha výrobních, a jiných, procesů. Například řídící jednotka může vyhodnotit informace ze zabezpečovacích zařízení jako zastaralé, a tudíž jim nebude přikládat adekvátní váhu. Vše závisí na konkrétním nastavení systému.

Vektor útoku Pro testovací účely byla změněna architektura laboratorní infrastruktury, zobrazené na obrázku 2.4 na architekturu zobrazenou na obrázku 2.7. Validní NTP server je spuštěn na Raspberry Pi s adresou `192.168.88.11/24`. Naše infikované zařízení bylo fyzicky umístěno do konfigurace MITM útoku, kdy byly dvě síťové karty spojeny do mostu.



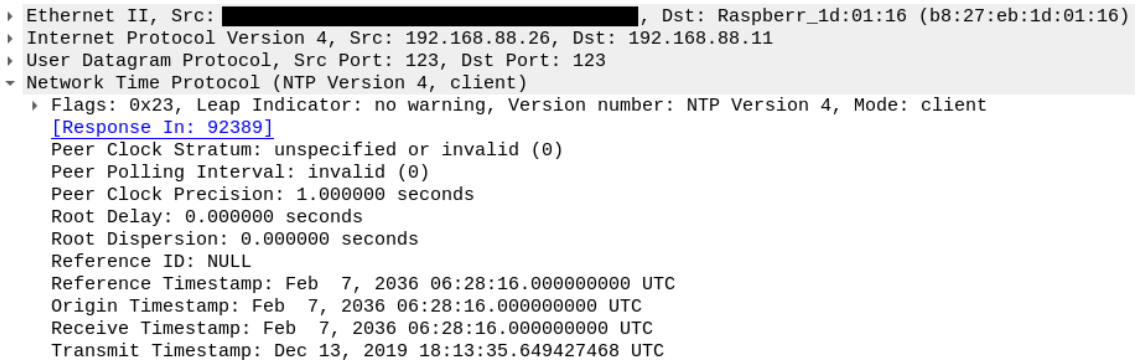
Obr. 2.7: Model testovací infrastruktury pro NTP spoofing.

Díky modifikaci testovací konfigurace odpadla nutnost provádět ARP (Address Resolution Protocol) spoofing a mohli jsme se soustředit pouze na samotnou modifikaci zachycených paketů. S pomocí knihovny `libtins`[2] napsal autor program pro zachytávání a podvrhování NTP paketů. Po spuštění program odposlouchává procházející komunikaci, z které odposlechne formát odpovědi validního NTP serveru. Po jistém počtu odpovědí začne utilita[3] aktivně vstupovat do konverzace. Zároveň dojde k odpojení komunikace z NTP serveru, aby nemohl odesílat validní zprávy. V momentě příchodu paketu na port 123 utilita rozpozná dotaz NTP klienta na server a odpoví předem připraveným ethernetovým rámcem, který vypadá jako rámec odeslaný z NTP serveru. Zpráva zapouzdřená ve zmíněném rámci má odpověď

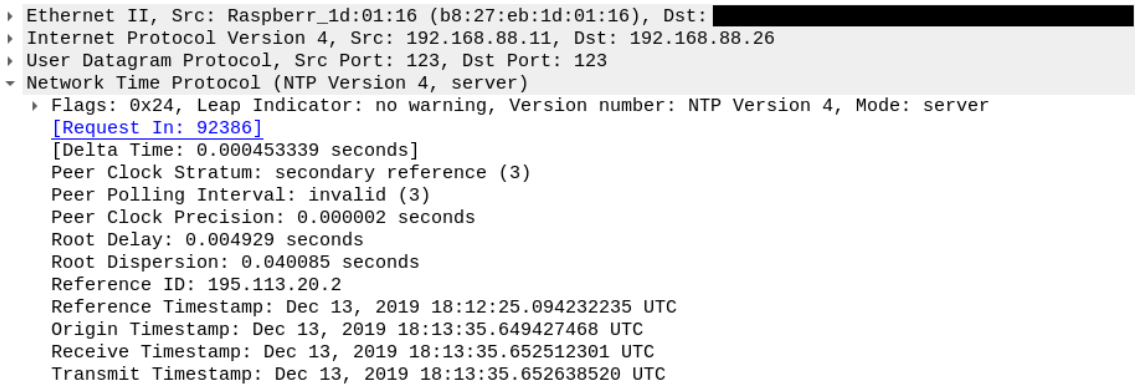
[2] https://github.com/mfontanini/libtins

[3] Jednoduchý nástroj vytvořený za jediným účelem

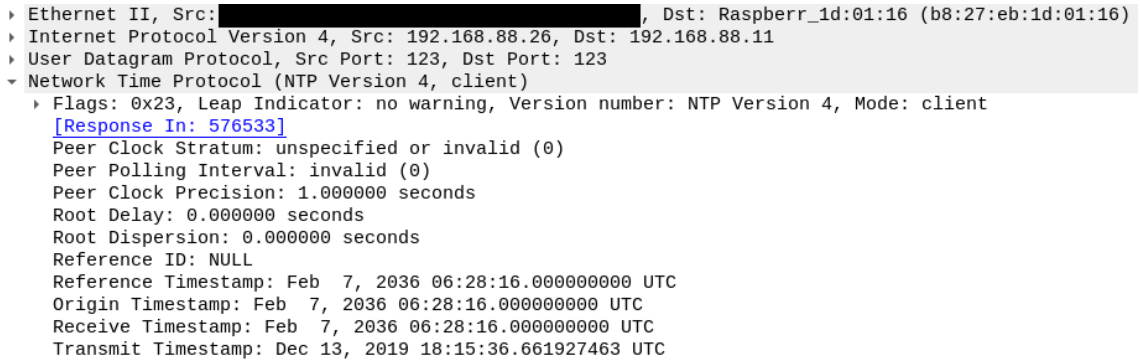
s podvrženým časem. Validní komunikace lze vidět na obrázcích 2.8 a 2.9. Naopak, podvržená komunikace lze vidět na obrázcích 2.10 a 2.11. Stejně jako v předchozím případě musíme cenzurovat identifikační údaje zařízení A. Můžeme si všimnout stejné MAC adresy odesílatele odpovědi v případě podvržené komunikace. Zachování této ethernetové hlavičky bylo možné z důvodu ruční konstrukce paketu na míru situaci.



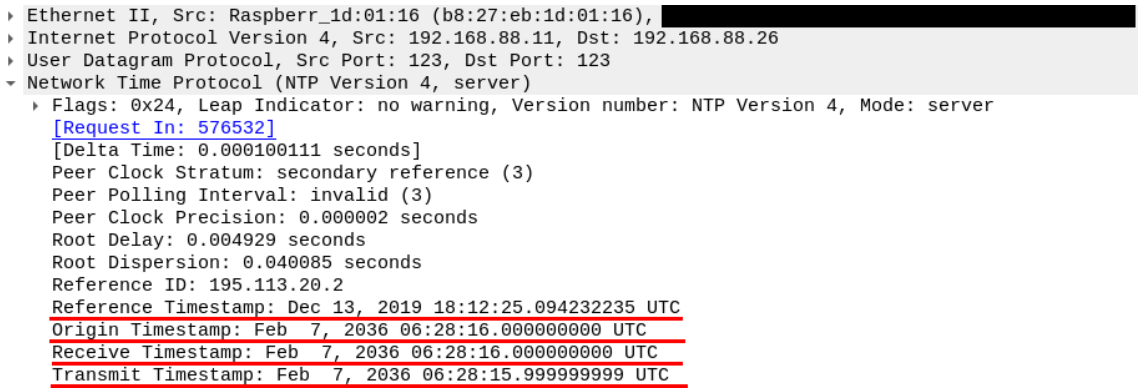
Obr. 2.8: Validní dotaz na NTP server.



Obr. 2.9: Validní odpověď NTP serveru.



Obr. 2.10: Validní dotaz klienta.



Obr. 2.11: Podvržená odpověď IT/OT uzlu.

Při testování daného útoku autor zjistil, že není zcela úspěšný, jelikož i přes potvrzení přijetí podvrženého času nedošlo k jeho převzetí a zařízení A si stále drželo svůj vnitřní čas pomocí vnitřního, bateriemi napájeného, zdroje. Tedy nedojde k vynulování času ani při výpadku napájení. Stalo se tak z několika důvodů. První důvod je definován v samotném standardu definujícím službu NTP [20]. Jedná se o mechanismus zabraňující nárazovým změnám času pomocí konstanty nazvané "práh paniky". V angličtině "panic threshold". V momentě přijetí paketu obsahující podvržený čas dojde k porovnání s vnitřním časem zařízení a v případě, že je rozdíl větší než hodnota prahu paniky, ve většině případů 1000 sekund, dojde k potvrzení přijetí, ale zařízení nepřenastaví své hodiny. Mechanismus se dá obejít pomocí pomalé inkrementace času. Principiálně se jedná o odesílání množství paketů každých 300 sekund, což je práh pro přijetí nového času, které postupně mění čas o hodnotu menší než práh paniky. Pro posunutí času o jeden rok, při posunutí jednou za 5 minut o 16 minut, bude trvat alespoň 114 dní [20].

Bohužel zmíněná metoda útoku na NTP klienta zařízení A není možná, jelikož na konec paketů je přidáván podpis specifikovaný ve standardu IEC 62351-6 pro protokol GOOSE a SMV. I v případě NTP klienta je každý NTP paket podepsán, čímž je zajištěna integrita a autenticita dat.

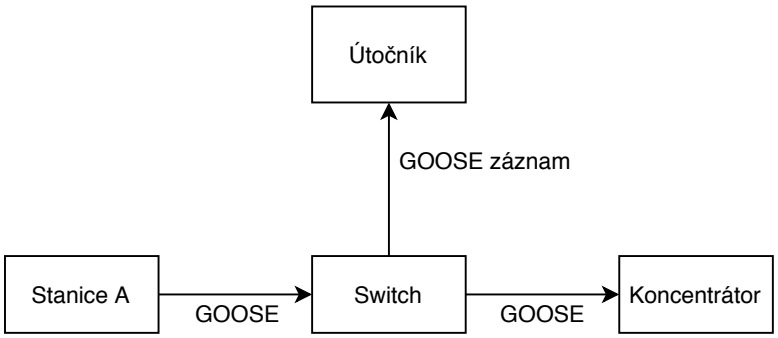
2.4.3 GOOSE replay

Útok GOOSE replay spočívá v zaznamenání přenášené komunikace mezi stanicemi a následném přehrání daného záznamu pro vyvolání chybového stavu systému.

Teoretický scénář Představme si následující situaci. Útočník sleduje, zachytává a analyzuje provoz v OT síti průmyslového zařízení. Studuje chování sítě a pozoruje výkyvy z normálu. Všimne si situace, která způsobila sepnutí ochranného mechanismu zařízení a tím zpomalení, případně zastavení výroby. Nyní útočník vidí automatickou reakci systému i reakci velení na daný vstup. Díky těmto informacím je schopen predikovat budoucí chování systému v podobné situaci. Například při vzniku kritického přepětí v síti může dojít k automatickému odpojení některých stanic, aby se zabránilo kaskádovému efektu. Při umělém vyvolání této reakce může dojít k vysokým finančním škodám.

Vektor útoku Útok probíhal na virtualizované infrastruktuře, kde autor předpokládá průnik útočníka k IT/OT uzlu a jeho ovládnutí, což mu umožní využít daný uzel jako pivot pro provedení útoku. Autor pro tento účel, jak je znázorněno na obrázku 2.12, zaznamenal pomocí nástroje Wireshark komunikaci mezi stanicí A

a koncentrátorem. Nyní má útočník (v našem případě autor) k dispozici záznam, který může přehrát zpět do sítě.



Obr. 2.12: Zaznamenání GOOSE paketů.

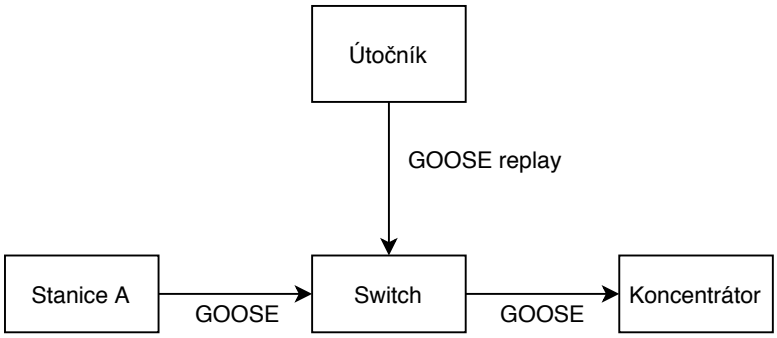
Abychom přiměli koncentrátor přijmout námi odeslána data, musí být splněno několik podmínek. Musí souhlasit sekvenční číslo, číslo indikující pořadí změny statusu a časová razítka indikující poslední změnu statusu, čas odeslání GOOSE paketu a samotnou MAC adresu odesílajícího zařízení, aby koncentrátor nepoznal, že do sítě data odesílá někdo jiný. Proto nestačí zaznamenaná data pouze znovu odeslat, ale musí dojít k jejich modifikaci. Zároveň musí být změněna MAC adresa pomocí utility `macchanger` na adresu stanice A a až pak může proběhnout spuštění útoku.

Pro tento účel napsal autor parser[4] protokolu GOOSE, který splňuje naše požadavky: umožňuje nám přečíst jednotlivá pole protokolu a následně modifikovat klíčová pole pro provedení útoku. Parser je napsán v jazyce C++, využívá pouze standardní knihovny jazyka C++ a C, takže jej může přeložit a spustit kdokoliv, kdo má nainstalovanou základní vývojářskou sadu. Podpora protokolu GOOSE je bohužel omezena pouze na data, která můžeme vidět v komunikaci mezi koncentrátorem a stanicí A, jelikož autor našel rozpor mezi definovanými konstantami TLV v provozu zachyceným zmíněnými stroji, a materiály popisující protokol GOOSE [17]. Autor plánuje tento parser dokončit a zveřejnit pod licencí GNU GPL v3. Podrobnější popis lze nalézt v sekci 2.5.

Program realizující útok postupně čte soubor, parsuje jednotlivé pakety, modifikuje číslo statusu, časové razítko indikující poslední změnu statusu, sekvenční číslo a čas odeslání paketu. Modifikovaný paket následně v daném časovém intervalu odešle na předem definovanou MAC adresu. Schéma lze vidět na obrázku 2.13.

Změny MAC adresy docílil autor pomocí utility `macchanger`, kdy nastavil MAC adresu útočícího stroje shodně s MAC adresou stanice A. Koncentrátoru se tedy jeví jako stanice A.

[4]Parser je nástroj sloužící k přečtení souboru či dat a jejich zpracování.



Obr. 2.13: Přehrání GOOSE paketů.

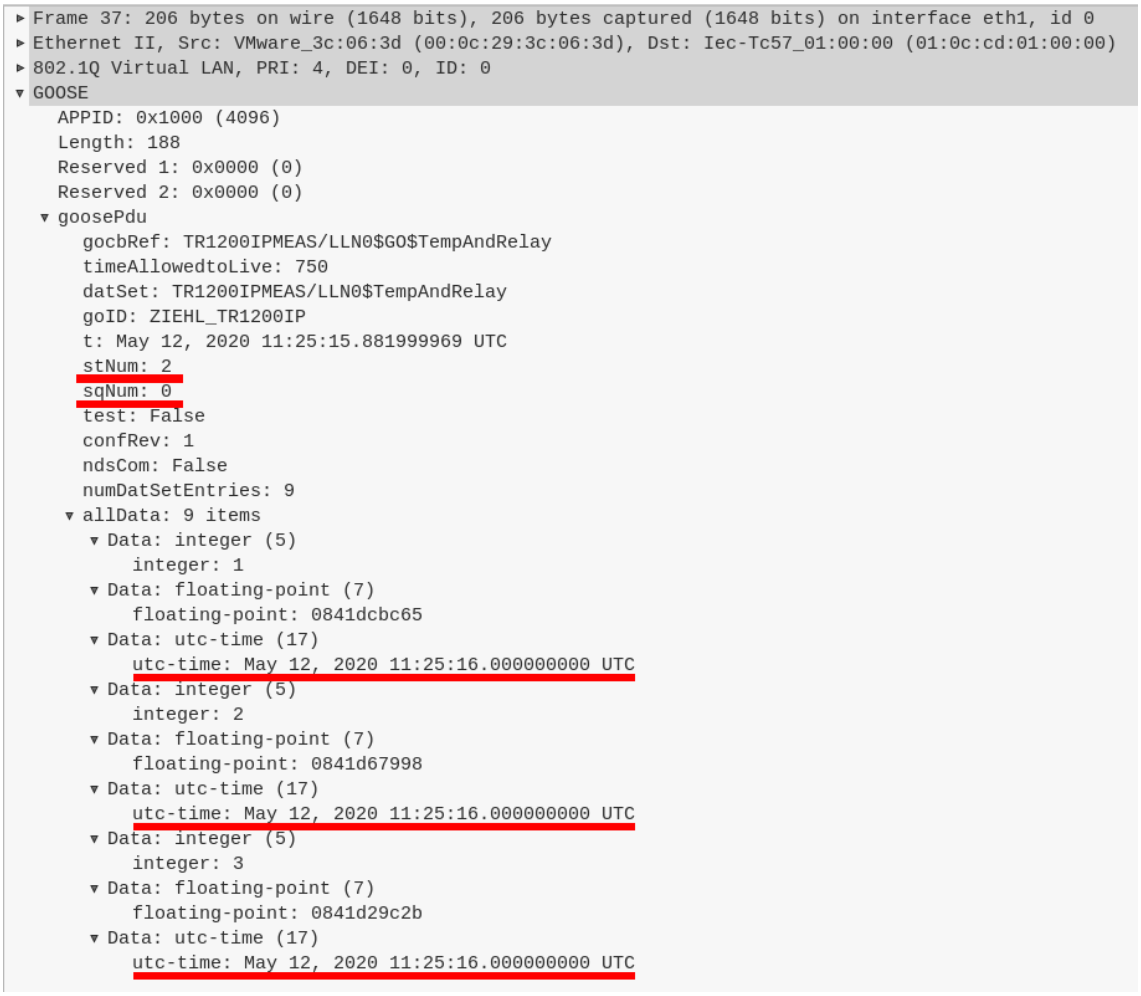
Průběh útoku lze vidět na obrázcích níže. Na obrázku 2.14 můžeme vidět validní paket odeslaný stanicí A. Povšimněme si adres v hlavičce ethernetového rámce, polí `stNum`, `sqNum` a položky `utc-time` v datové části protokolu GOOSE.

```
▸ Frame 36: 206 bytes on wire (1648 bits), 206 bytes captured (1648 bits) on interface eth1, id 0
▸ Ethernet II, Src: VMware_3c:06:3d (00:0c:29:3c:06:3d), Dst: Iec-Tc57_01:00:00 (01:0c:cd:01:00:00)
▸ 802.1Q Virtual LAN, PRI: 4, DEI: 0, ID: 0
▾ GOOSE
    APPID: 0x1000 (4096)
    Length: 188
    Reserved 1: 0x0000 (0)
    Reserved 2: 0x0000 (0)
  ▾ goosePdu
      gocbRef: TR1200IPMEAS/LLN0$GO$TempAndRelay
      timeAllowedtoLive: 750
      datSet: TR1200IPMEAS/LLN0$TempAndRelay
      goID: ZIEHL_TR1200IP
      t: May 12, 2020 11:25:08.813999950 UTC
      stNum: 1
      sqNum: 10
      test: False
      confRev: 1
      ndsCom: False
      numDatSetEntries: 9
    ▾ allData: 9 items
      ▾ Data: integer (5)
          integer: 1
      ▾ Data: floating-point (7)
          floating-point: 0841bbeca9
      ▾ Data: utc-time (17)
          utc-time: Apr  1, 2019 21:37:00.000000000 UTC
      ▾ Data: integer (5)
          integer: 2
      ▾ Data: floating-point (7)
          floating-point: 0841b54d23
      ▾ Data: utc-time (17)
          utc-time: Apr  1, 2019 21:37:00.000000000 UTC
      ▾ Data: integer (5)
          integer: 3
      ▾ Data: floating-point (7)
          floating-point: 0841b1ca53
      ▾ Data: utc-time (17)
          utc-time: Apr  1, 2019 21:37:00.000000000 UTC
```

Obr. 2.14: Validní GOOSE paket stanice A.

Nyní srovnáme tato pole s daty podvrženého paketu, který vidíme na obrázku 2.15.

Můžeme si všimnout, že zdrojové a cílové adresy jsou shodné v obou paketech, údaje o VLAN taktéž. Ovšem pole `stNum` a `sqNum` mají jiné hodnoty, které byly nastaveny autorovým programem. Položky `utc-time` taktéž.



Obr. 2.15: Paket útoku GOOSE replay.

Díky tomu koncentrátor považuje podvržená data za validní a aktuální, proto je bez potíží přijme. Totéž se ovšem nedá říci o následujícím validním paketu, který má menší `stNum`, tudíž jej koncentrátor zamítne, jak můžeme vidět na výpisech 2.2 a 2.3.

Výpis 2.2: Přijetí podvrženého paketu koncentrátorem.

```
TR1200IP GOOSE, stNum: 1, sqNum: 10, delta t: 500
  Sensor: 1
  Temperature: 23.4906 °C
  Timestamp: 01/04/2019 23:37:00
  Sensor: 2
  Temperature: 22.6627 °C
  Timestamp: 01/04/2019 23:37:00
  Sensor: 3
  Temperature: 22.2238 °C
    Timestamp: 01/04/2019 23:37:00

TR1200IP GOOSE, stNum: 2, sqNum: 0, delta t: 0
  Sensor: 1
  Temperature: 27.592 °C
  Timestamp: 12/05/2020 13:25:16
  Sensor: 2
  Temperature: 26.8094 °C
  Timestamp: 12/05/2020 13:25:16
  Sensor: 3
  Temperature: 26.3263 °C
  Timestamp: 12/05/2020 13:25:16
```

Výpis 2.3: Zamítnutí validního paketu.

```
TR1200IP GOOSE, stNum: 2, sqNum: 4, delta t: 100
  Sensor: 1
  Temperature: 30.3355 °C
  Timestamp: 12/05/2020 13:25:16
  Sensor: 2
  Temperature: 30.8806 °C
  Timestamp: 12/05/2020 13:25:16
  Sensor: 3
  Temperature: 30.2912 °C
  Timestamp: 12/05/2020 13:25:16

Warning - GOOSE frame values for APPID 0x1000 differ
   from saved values!
  Last accepted - stNum: 2, sqNum: 4
  Received      - stNum: 1, sqNum: 11
```

2.4.4 GOOSE flood

Při využití útoku GOOSE flood jde o co největší zahlcení sítě a jednotlivých členů komunikujících protokolem GOOSE. Dochází k nárůstu potřebného výkonu pro zpracování dat a tím zpomalení potřebné reakce systému k odvrácení hrozby, či dokonce ke kolapsu sítě z důvodu nedostatečné přenosové kapacity.

Teoretický scénář Mějme následující situaci. Vzhledem k pomalému nasazování aktuálních technologií v průmyslové sféře se může stát, že průmyslové zařízení, na které útočník chce zaútočit, nebude mít nejmodernější gigabitovou síť, avšak pouze sto megabitovou síť. V takovém případě může dojít k rychlému zahlcení sítě a nucenému zahazování paketů, které může být v krizové situaci nepřijatelné z důvodu zpomalení či až zastavení odezvy systému. Nejen automatické, ale i odezvy z velínu, jelikož nebude fungovat síťové spojení. Útočník tak získá drahocenný čas pro provedení veškerých zamýšlených útoků, aniž by se musel obávat okamžitého sepnutí bezpečnostních mechanismů.

Vektor útoku Předpokládejme průnik útočníka do infrastruktury průmyslového zařízení a ovládnutí IT/OT uzlu, který útočník využije jako pivot pro útoky na zařízení v OT síti.

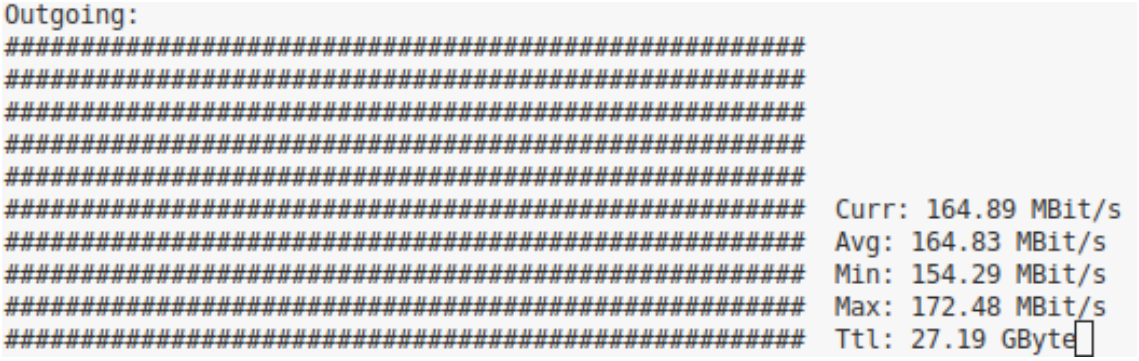
Pro úspěšné vytížení sítě a uzlu musíme posílat validní pakety protokolu GOOSE. Abychom tohoto dosáhli, musíme odesílat správné `stNum` a správné časové razítko. Pokud bychom tak neučinili, síť bude zahlcena, avšak koncentrátor pakety místo jejich zpracování zahodí a ušetří výpočetní výkon, což je pro útočníka nežádoucí. Nejdříve ale musíme změnit MAC adresu pomocí utility `macchanger` na adresu stanice A a až pak můžeme spustit samotný útok.

K provedení útoku můžeme přistoupit několika způsoby. První způsob odesílá co nejvíce paketů s velkým objemem dat a druhý odesílá co nejvíce paketů s minimálním množstvím datových položek. Každé má své výhody a nevýhody.

Nejdříve se budeme zabývat prvním zmíněným způsobem. Vycházíme z faktu, že zpracování paketu vyžaduje procesorový čas. Proto vygenerujeme pakety, které obsahují 100 datových položek. Velikost takového paketu je 850 B (byte). Na obrázku 2.16 lze vidět zmíněný paket zachycený programem Wireshark. Zleva doprava čísla znamenají: číslo zachyceného paketu, čas zachycení paketu od začátku zachytávání, zdrojovou MAC adresu, cílovou MAC adresu, protokol a velikost v B (bytech).

Obr. 2.16: Paket nesoucí 100 položek v datové části.

Na obrázku 2.17 můžeme vidět velikost generovaného provozu při využití velkého objemu dat v datové části protokolu GOOSE. Utilita `nload` ukazuje 164.89 Mbit (megabitů) za sekundu. Generování se zdá být dost rychlé, avšak v gigabitových sítích je to pro zahlcení zcela nedostačující. Nízká rychlost je způsobená dílčím alokováním a přiřazováním paměti během generování paketů. Abychom vyřešili zmíněný problém, přikročil autor k druhému způsobu, kdy generuje mnoho paketů s malým počtem datových polí v datové části.



Obr. 2.17: Rychlost provozu generovaných paketů se 100 datovými položkami.

Vzhledem k náročnosti alokace paměti pro přidávání datových položek do paketů se autor rozhodl snížit počet položek ze sta na jednu. Generování paketů se podstatně zrychlilo. Tím došlo k většímu zatížení sítě, přičemž vytížení koncentrátoru bylo takové, že koncentrátor nestíhal data zpracovávat a, jak vidíme ve výpisu 2.4, musel je zahazovat. Můžeme vidět, že koncentrátor byl donucen zahodit některé pakety, jelikož jeho buffery byly přeplněny. Důsledkem je rozbití schopnosti přijímat přicházející data z důvodu nesouhlasícího `stNum`. Díky vysokému `stNum` dojde také k zahazování validních paketů s nižším `stNum`. Na podobném principu funguje útok GOOSE stNum, který si popíšeme v další části této kapitoly.

Výpis 2.4: Výstup koncentrátoru po zahlcení vstupních bufferů.

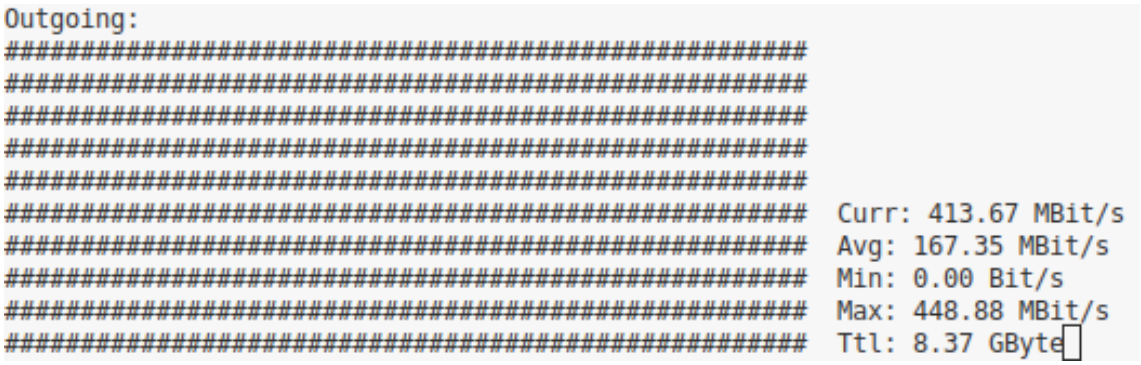
```
TR1200IP GOOSE, stNum: 48767, sqNum: 5, delta t: 0
TR1200IP GOOSE, stNum: 48767, sqNum: 6, delta t: 0

Warning - GOOSE frame values for APPID 0x1000 differ
   from saved values!
  Last accepted - stNum: 48767, sqNum: 6
  Received      - stNum: 48792, sqNum: 3

Warning - GOOSE frame values for APPID 0x1000 differ
   from saved values!
```

```
Last accepted - stNum: 48767, sqNum: 6
Received      - stNum: 48792, sqNum: 4
```

Na obrázku 2.18 můžeme vidět rozdíl ve velikosti jednotlivých paketů. Oproti předchozímu přístupu má celý paket pouze 154 B. Stejně jako u minulého obrázku z programu Wireshark, první je číslo paketu, druhý čas zachycení od začátku zachytávání, následuje zdrojová a cílová MAC adresa, protokol a velikost.

Obr. 2.18: Paket nesoucí pouze jednu položku v datové části.

Oproti předchozí formě útoku nyní dokáže útočník generovat násobně větší objem dat. Při testování útoku s jedinou instancí programu naměřil autor rychlosti okolo 400 megabitů za sekundu, což můžeme vidět na obrázku 2.19. Nízká průměrná rychlost je způsobena prodlevou mezi spuštěním utility `nload` a zahájením útoku.



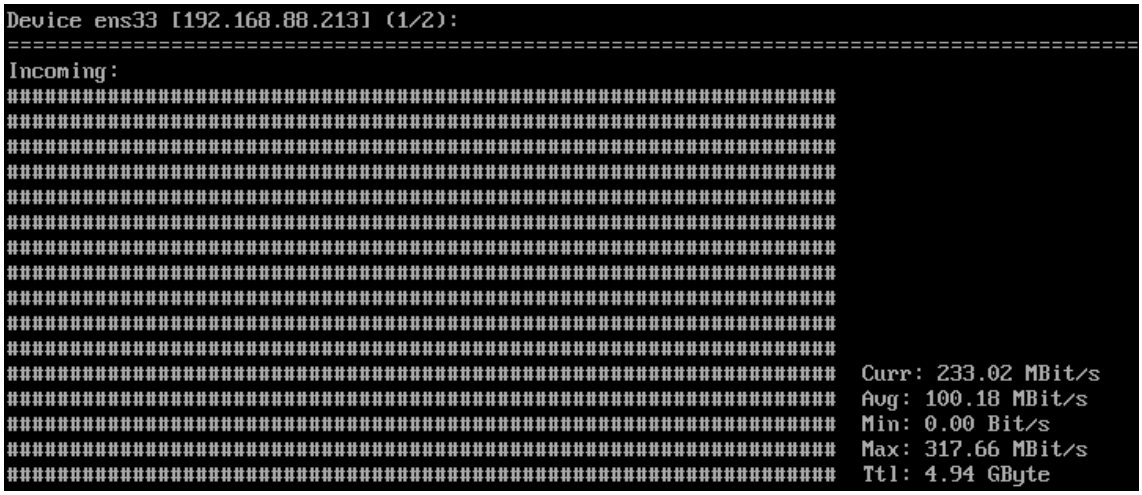
Obr. 2.19: Rychlost generovaného provozu s malým množstvím dat.

Jak vidíme na předchozím obrázku, generovaný objem je násobně vyšší. Na zahlcení sítě to sice nestačí, ale stačí to na zahlcení síťového rozhraní koncentrátoru. Jak můžeme vidět na obrázku 2.20, generovaný objem dat s hodnotou násobně překračuje schopnost koncentrátoru přijímat data ze sítě už jen kvůli jeho neschopnosti parsovat data, jelikož procesor je vytížený na 100 %. Výstup je z aplikace `htop`.

Obr. 2.20: Vytížení procesoru koncentrátoru.

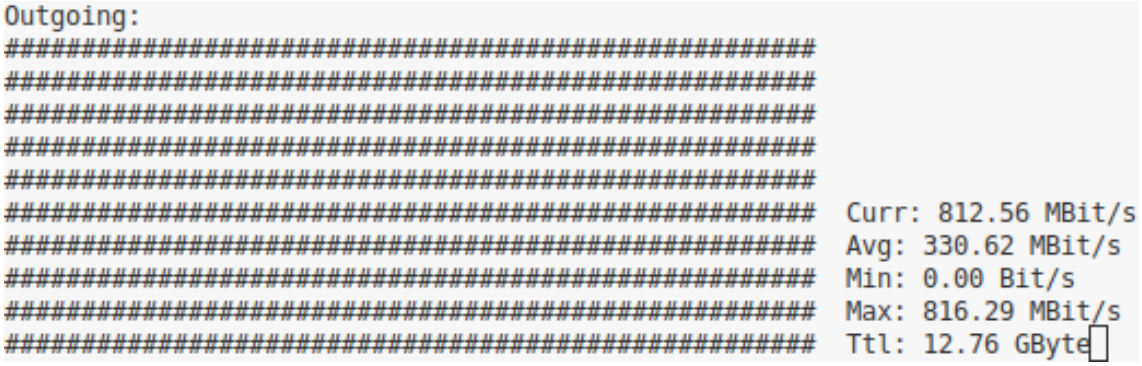
Nejenom, že dochází ke stoprocentnímu vytížení procesoru, dochází i k neschopnosti zpracovávat přijímaná data z důvodu nízké kapacity vstupního rozhraní. Jak

můžeme vidět na obrázku 2.21, generovaný objem dat je vyšší, než je objem, který je schopen koncentrátor přijmout pro zpracování. Nízká hodnota průměru je způsobena prodlevou mezi zahájením měření a spuštěním útoku.



Obr. 2.21: Rychlost vstupu s malým množstvím dat do koncentrátoru.

I když se povedlo zahltit samotný koncentrátor, bohužel se nepovedlo vytížit síť. Naštěstí je GOOSE flood útok jednoduše škálovatelný a můžeme spustit více instancí útoku na jednom stroji. Na obrázku 2.22 lze vidět zdvojnásobení objemu generovaného provozu, což je s hodnotou okolo 800 megabitů za sekundu značný rozdíl. Stejně jako v předchozích ukázkách je nízká průměrná hodnota dána prodlevou mezi spuštěním útoků a zahájením měření.

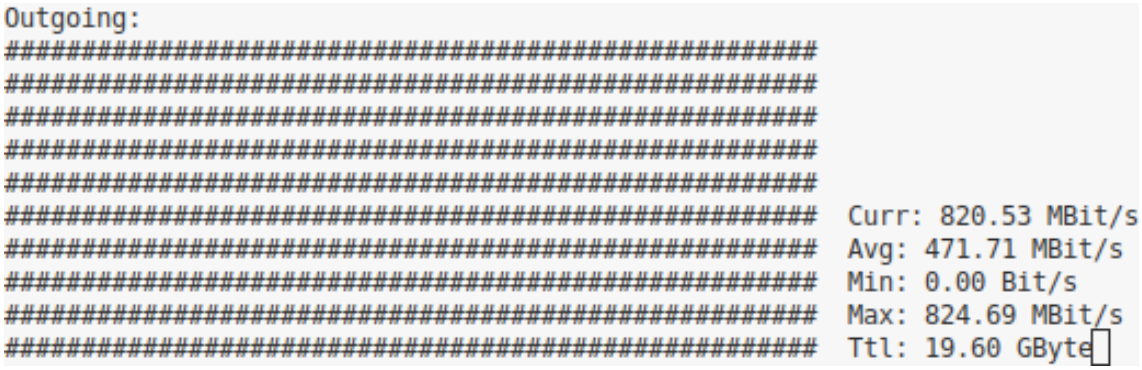


Obr. 2.22: Objem generovaných dat při spuštění dvou instancí útoku.

Jak lze vidět na obrázku 2.23, při škálování útoku zůstává objem přijímaných dat stejný, takže na koncentrátor nemá vliv. Stejně jako na předchozích obrázcích je průměrná hodnota malá kvůli časovému rozdílu mezi zahájením měření a spuštěním útoků.

Obr. 2.23: Objem koncentrátorem přijímaných dat při spuštění dvou instancí útoku.

Při dalším škálování na jednom procesoru je rozdíl mezi množstvím generovaných dat malý z důvodu zatížení síťového rozhraní, které musí dělit zdroje mezi vstupní provoz do koncentrátoru a výstupní z hostitelského stroje. Lze vidět na obrázku 2.24. Při využití více počítačů ke generování provozu pro útok dojde k dostatečnému navýšení objemu dat pro vytížení sítě.



Obr. 2.24: Objem generovaných dat při spuštění tří instancí útoku.

2.4.5 GOOSE stNum

GOOSE stNum útok slouží k převzetí kontroly nad vstupem do zařízení pomocí vnucení vysokého indikátoru změny stavu, přičemž zařízení bude odmítat příchozí pakety s nižší hodnotou indikátoru změny stavu a dojde k narušení reakcí systému na krizové události.

Teoretický scénář Mějme průmyslové zařízení obsahující transformátor, sondu měřící jeho teplotu, koncentrátor a aktuátor ovládající daný transformátor. Pro jednoduchost příkladu má aktuátor stavy zapnuto a vypnuto. Měřící sonda komunikuje s koncentrátorem protokolem GOOSE a koncentrátor v případě kritické teploty transformátoru pošle signál aktuátoru, který transformátor odpojí. Pomocí útoku mířeného na zneužití čítače změn stavu útočník zabrání koncentrátoru přijímat data ze sondy a koncentrátor nebude moci zareagovat na kritický stav. Kvůli tomu dojde k překročení maximální provozní teploty transformátoru, které může skončit i požárem, či může dojít k významným finančním i jiným škodám.

Vektor útoku Útok probíhal na virtualizované infrastruktuře, kde se předpokládá průnik útočníka a ovládnutí IT/OT uzlu. V ten moment útočník začne skenovat provoz a připraví si GOOSE stNum útok. Jako pro všechny útoky na protokol GOOSE byla i zde využita knihovna libiec61850.

Abychom přiměli koncentrátor ignorovat validní zprávy sondy, musíme mu odeslat zprávu, která bude mít inkrementovaný čítač změny stavu `stNum`. Nejdříve dojde ke změně MAC adresy pomocí utility `macchanger` na adresu stanice a pak proběhne spuštění útoku. Posléze autorem napsaný program odposlechne aktuální hodnotu `stNum`, inkrementuje ji o jednu jednotku a začne odesílat pakety, přičemž validně inkrementuje čítače `stNum` a `sqNum`. Zmíněným chováním aplikace udržuje koncentrátor ve stavu odmítání validní pakety ze stanice. Ve výpisu 2.5 můžeme vidět potvrzení koncentrátoru o přijetí paketu s inkrementovaným čítačem `stNum`. Ačkoliv podvržený paket nepřenáší žádné údaje z teploměru, koncentrátor si pamatuje staré, které vytiskne do záznamu.

Výpis 2.5: Přijetí podvrženého paketu od útočníka s čítačem `stNum`.

```
TR1200IP GOOSE, stNum: 1, sqNum: 8, delta t: 501
  Sensor: 1
  Temperature: 24.3563 °C
  Timestamp: 01/04/2019 21:37:00
  Sensor: 2
  Temperature: 27.3625 °C
  Timestamp: 01/04/2019 21:37:00
```

```
  Sensor: 3
  Temperature: 27.2406 °C
  Timestamp: 01/04/2019 21:37:00


TR1200IP GOOSE, stNum: 2, sqNum: 0, delta t: 0
  Sensor: 1
  Temperature: 24.3563 °C
  Timestamp: 01/04/2019 21:37:00
  Sensor: 2
  Temperature: 27.3625 °C
  Timestamp: 01/04/2019 21:37:00
  Sensor: 3
  Temperature: 27.2406 °C
  Timestamp: 01/04/2019 21:37:00
```

Jak již bylo řečeno výše, a jak můžeme vidět ve výpisu 2.6, podvržené pakety nepřenáší žádná data ze stanice.

Výpis 2.6: Zamítavá reakce koncentrátoru po přijetí validního `stNum`.

```
TR1200IP GOOSE, stNum: 2, sqNum: 4, delta t: 99
  Sensor: 1
  Temperature: 24.3563 °C
  Timestamp: 01/04/2019 21:37:00
  Sensor: 2
  Temperature: 27.3625 °C
  Timestamp: 01/04/2019 21:37:00
  Sensor: 3
  Temperature: 27.2406 °C
  Timestamp: 01/04/2019 21:37:00

Warning - GOOSE frame values for APPID 0x1000 differ
   from saved values!
  Last accepted - stNum: 2, sqNum: 4
  Received      - stNum: 1, sqNum: 7
```

V této konkrétní implementaci útoku jsou podvržené pakety odesílány každých 100 milisekund abychom měli jistotu, že validní data, která jsou odesílána cca. každých 0,5 sekundy, nebudou mít vyšší hodnotu čítače `stNum`.

Na obrázku 2.25 může čtenář vidět záznam probíhajícího útoku, kdy mezi dva validními pakety o velikosti 206 bytů byly odeslány podvržené pakety o velikosti 145

bytů. Následně lze na obrázcích 2.26 a 2.27 můžeme vidět obsah validního, respektive podvrženého paketu.

No.	Time	Source	Destination	Protocol	Length
1	0.000000000	VMware_3c:06:3d	Iec-Tc57_01:00:00	GOOSE	206
2	0.000549471	VMware_3c:06:3d	Iec-Tc57_01:00:00	GOOSE	145
3	0.100689049	VMware_3c:06:3d	Iec-Tc57_01:00:00	GOOSE	145
4	0.200744489	VMware_3c:06:3d	Iec-Tc57_01:00:00	GOOSE	145
5	0.300870347	VMware_3c:06:3d	Iec-Tc57_01:00:00	GOOSE	145
6	0.400963447	VMware_3c:06:3d	Iec-Tc57_01:00:00	GOOSE	145
7	0.499980563	VMware_3c:06:3d	Iec-Tc57_01:00:00	GOOSE	206

Obr. 2.25: Záznam síťového provozu během útoku GOOSE stNum.

```
▾ GOOSE
    APPID: 0x1000 (4096)
    Length: 188
    Reserved 1: 0x0000 (0)
    Reserved 2: 0x0000 (0)
  ▾ goosePdu
      gocbRef: TR1200IPMEAS/LLN0$GO$TempAndRelay
      timeAllowedtoLive: 750
      datSet: TR1200IPMEAS/LLN0$TempAndRelay
      goID: ZIEHL_TR1200IP
      t: May 13, 2020 18:58:12.077999949 UTC
      stNum: 1
      sqNum: 8
      test: False
      confRev: 1
      ndsCom: False
      numDatSetEntries: 9
    ▸ allData: 9 items
```

Obr. 2.26: Validní pakety obsahující data.

```
▾ GOOSE
    APPID: 0x1000 (4096)
    Length: 127
    Reserved 1: 0x0000 (0)
    Reserved 2: 0x0000 (0)
  ▾ goosePdu
      gocbRef: TR1200IPMEAS/LLN0$GO$TempAndRelay
      timeAllowedtoLive: 750
      datSet: TR1200IPMEAS/LLN0$TempAndRelay
      goID: ZIEHL_TR1200IP
      t: May 13, 2020 18:58:16.121999979 UTC
      stNum: 2
      sqNum: 0
      test: False
      confRev: 1
      ndsCom: False
      numDatSetEntries: 0
      allData: 0 items
```

Obr. 2.27: Podvržené pakety bez dat.

2.4.6 GOOSE semantic

Použití útoku GOOSE semantic přichází na řadu v případě, že chceme přimět koncentrátor k reakci na námi zvolené podněty systému. Pomocí podvržených zpráv v datové části paketu může útočník přimět koncentrátor ke změně svého chování a v případě kombinace s jinými útoky vyvolat libovolnou systémovou reakci, aniž by kdokoliv byl schopen reagovat.

Teoretický scénář Představme si stejnou situaci jako v předchozím příkladu. Avšak s drobným rozdílem. Útočník nechce docílit zničení transformátoru, ale chce jej bez poškození vypnout. Aby toho docílil, napadne IT/OT uzel a začne posílat zprávy s podvrženým obsahem. Spolu s podvrženým obsahem musí odeslat správnou kombinací stavového čítače a sekvenčního čísla, `stNum` a `sqNum`, tedy využít podporu útoku GOOSE stNum. Když takto vytvořené pakety pošle koncentrátoru ovládajícímu aktuátor transformátoru, může ovládat jeho výstupy a reakce. Pomocí výše zmíněného postupu dokáže transformátor například vypnout, a způsobit škody ve výrobním procesu. V lepším případě dojde pouze k finančním a materiálním ztrátám, v horším ke škodám na životech.

Vektor útoku Útok, jako v předchozích případech, začíná ovládnutím IT/OT uzlu, který útočník využije jako pivot. Autor na základě zachycené komunikace mezi stanicí A a koncentrátorem implementoval konfigurovatelnou aplikaci, která odesílá validně tvářící se pakety s podvrženými údaji v datové části. V kombinaci s útokem GOOSE stNum nejenže zabrání přijímání validních paketů ze stanice A, ale pozmění chování koncentrátoru a přiměje jej k provedení zamýšlené akce. Po změně MAC adresy pomocí utility `macchanger` na adresu stanice A proběhne spuštění útoku. Po spuštění útoku aplikace odposlechne komunikaci mezi stanicí A a koncentrátorem a nastaví čítač `stNum` na přečtenou hodnotu plus jedna. Tím zabráníme koncentrátoru přijímat validní pakety, které jsou stále odesílány nezávisle na našem útoku. Nyní bude koncentrátor přijímat jakoukoliv sadu námi vymyšlených hodnot.

Reakce koncentrátoru je názorně ukázána na výpisu z logovacího souboru. Ve výpisu 2.7 lze vidět validní paket, po němž následuje autorem odeslaný podvržený paket v rámci útoku. Informaci o nevalidním paketu můžeme zanedbat, jelikož stanice A odesílá originální paket a ihned vzápětí i jeho duplikát. Koncentrátor pak do souboru zapisuje informace o nevalidním paketu.

Výpis 2.7: Reakce koncentrátoru na podvržený paket

```
TR1200IP GOOSE, stNum: 1, sqNum: 4, delta t: 499
  Sensor: 1
  Temperature: 29.1581 °C
  Timestamp: 01/04/2019 27:37:00
  Sensor: 2
  Temperature: 29.9597 °C
  Timestamp: 01/04/2019 27:37:00
  Sensor: 3
  Temperature: 30.3778 °C
  Timestamp: 01/04/2019 27:37:00

TR1200IP GOOSE, stNum: 2, sqNum: 0, delta t: 100
  Sensor: 1337
  Temperature: 1337 °C
  Timestamp: 14/05/2020 00:54:43
  Sensor: 1337
  Temperature: 1337 °C
  Timestamp: 14/05/2020 00:54:43
  Sensor: 1337
  Temperature: 1337 °C
  Timestamp: 14/05/2020 00:54:43
```

Narozdíl od předchozí situace, kdy nejdříve přišel validní a pak podvržený paket, nyní přichází pakety v opačném pořadí, tedy nejdříve podvržený a jako druhý validní paket. Koncentrátor validní paket zamítnul. Pozorný čtenář si všimne návaznosti validních paketů, kdy čítač změn stavů zůstává stále na hodnotě jedna, avšak sekvenční číslo se zvýšilo z hodnoty čtyři na hodnotu pět. Situaci můžeme vidět ve výpisu 2.8.

Výpis 2.8: Reakce koncentrátoru na validní paket po přijetí podvrženého paketu.

```
TR1200IP GOOSE, stNum: 2, sqNum: 4, delta t: 90
  Sensor: 1337
  Temperature: 1337 °C
  Timestamp: 14/05/2020 00:54:43
  Sensor: 1337
  Temperature: 1337 °C
  Timestamp: 14/05/2020 00:54:43
  Sensor: 1337
  Temperature: 1337 °C
```

```
    Timestamp: 14/05/2020 00:54:43

  Warning - GOOSE frame values for APPID 0x1000 differ
     from saved values!
    Last accepted - stNum: 2, sqNum: 3
    Received      - stNum: 1, sqNum: 5
```

Srovnání velikostí validního a podvrženého paketu můžeme vidět na obrázku 2.28. První paket z dvojice je validní. Jak může čtenář vidět, paket má velikost 206 B, zatímco paket podvržený je s celkovou délkou 209 B o tři byty delší. Pro naši implementaci koncentrátoru tento velikostní rozdíl nemá význam, avšak byla by potřeba identická velikost, musí se snížit identifikační čísla senzorů. Zde jsou to pro názornost čísla s šířkou 2 B.

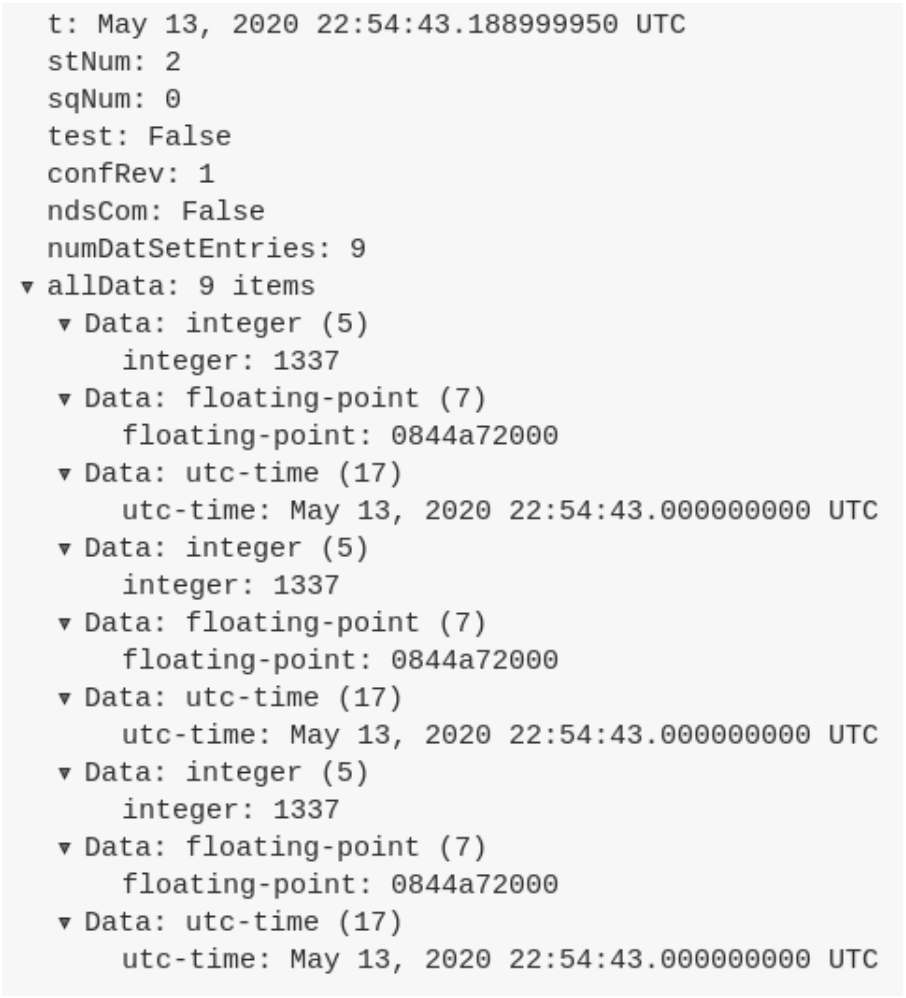
No.	Time	Source	Destination	Protocol	Length
1	0.000000000	VMware_3c:06:3d	Iec-Tc57_01:00:00	GOOSE	206
2	0.100405523	VMware_3c:06:3d	Iec-Tc57_01:00:00	GOOSE	209

Obr. 2.28: Srovnání velikostí validního a podvrženého paketu.

Na obrázcích 2.29 a 2.30 můžeme vidět po sobě jdoucí validní, respektive podvržený paket, přičemž si čtenář zajisté všimne změn v položkách `stNum`, `sqNum`, identifikátorech senzorů a jejich teplot stejně jako časová razítka naměřených teplot.

```
  t: May 13, 2020 22:54:41.045999944 UTC
  stNum: 1
  sqNum: 4
  test: False
  confRev: 1
  ndsCom: False
  numDatSetEntries: 9
▾ allData: 9 items
  ▾ Data: integer (5)
      integer: 1
  ▾ Data: floating-point (7)
      floating-point: 0841e943b5
  ▾ Data: utc-time (17)
      utc-time: Apr  1, 2019 15:37:00.000000000 UTC
  ▾ Data: integer (5)
      integer: 2
  ▾ Data: floating-point (7)
      floating-point: 0841efad83
  ▾ Data: utc-time (17)
      utc-time: Apr  1, 2019 15:37:00.000000000 UTC
  ▾ Data: integer (5)
      integer: 3
  ▾ Data: floating-point (7)
      floating-point: 0841f305ab
  ▾ Data: utc-time (17)
      utc-time: Apr  1, 2019 15:37:00.000000000 UTC
```

Obr. 2.29: Časové razítko, stNum a datová část validního paketu.

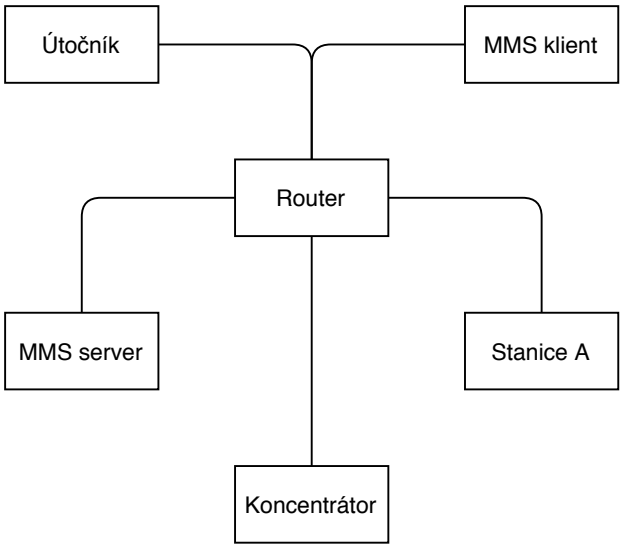


Obr. 2.30: Časové razítko, stNum a datová část podvrženého paketu.

2.4.7 MMS password capture

Útok MMS password capture, narozdíl od předchozích útoků na protokol GOOSE, necílí na OT síť, ale na IT síť. Jedná se o spuštění falešného MMS serveru, ke kterému se poté budou připojovat klienti v domnění, že se jedná o pravý server na, který jsou napojena IED v OT síti. V případě autentizace pomocí hesla útočník zjistí heslo k serveru, a poté jej může ovládnout.

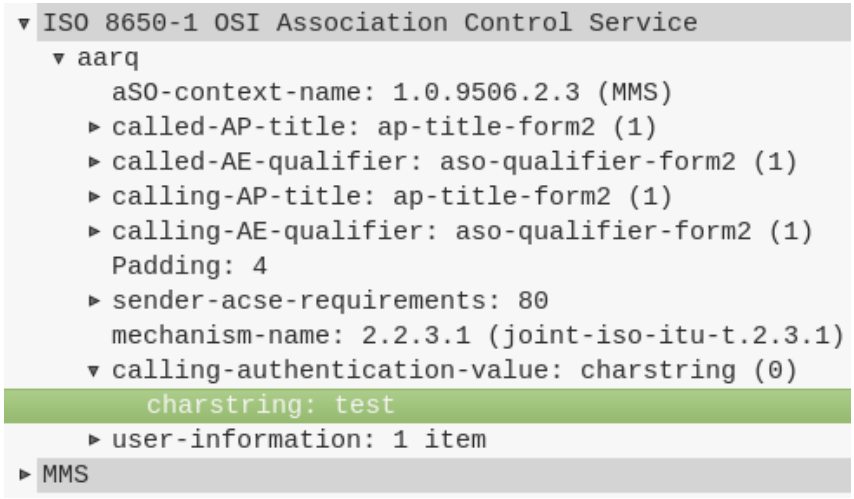
Teoretický scénář Využijeme modelové situace z předchozího útoku, kde máme transformátor, sondu, koncentrátor, a aktuátor. Zmíněnou sestavu doplníme o MMS server, který umožňuje správcům infrastruktury infrastrukturu sledovat a řídit. Stejně jako v předchozích situacích útočník pronikne do sítě a ovládne IT/OT uzel. Po prostudování topologie sítě zjistí, že se v síti nachází MMS klient a MMS server spravující VMD. Příklad topologie můžeme vidět na obrázku 2.31. IP adresy z důvodu běžící služby DHCP nepotřebujeme.



Obr. 2.31: Topologie modelové sítě.

Vektor útoku Vzhledem k zapouzdření protokolu MMS do struktury TCP/IP lze na protokol MMS provádět stejné útoky jako na jiné protokoly zapouzdřené do TCP/IP. Pokud klient a server nepoužívá šifrovanou komunikaci, útočník jednoduše přečte heslo z inicializačních paketů odesílaných klientem. Jak můžeme vidět na obrázku 2.32, heslo je zapouzdřeno do ISO 8650-1 OSI Association Control Service. Avšak v případě využití protokolu TLS bude komunikace zašifrována a útočník nebude mít možnost hesla jednoduše vyčíst ze zachycené komunikace.

Pro případ využití šifrované komunikace napsal autor program tvářící se jako server. Po spuštění program otevře port 102, což je výchozí port pro komunikaci MMS



Obr. 2.32: Heslo v textovém formátu.

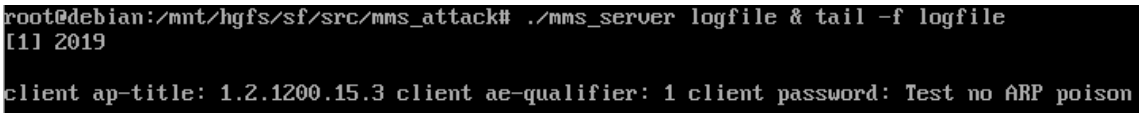
protokolem, a čeká na příchozí spojení od klienta. Aby program vypadal, že ověřuje heslo klienta a dělá to co má, je v hlavní smyčce programu nastaveno uspání procesu na jednu sekundu. Nová spojení jsou zpracovávána pomocí asynchronního volání funkcí knihovny libiec61850. Po navázání TCP spojení klientem klient odešle své přihlašovací údaje. Server přihlašovací údaje zpracuje a do záznamového souboru zapíše klientovo `ap-title`, `ae-qualifier` a jeho heslo. Útočník tak je tak schopen vytvořit seznam klientů a jejich hesel. Jednotlivé položky pak může využít k útoku na pravý server. Aby útok mohl být úspěšný, musí útočník provést Man in the Middle útok, jinak se nepodaří přesměrovat komunikaci z pravého serveru na falešný.

Pro testovací účely je jako validní server zvolena stejná aplikace jako pro podvržený server. Zvolený způsob testování nemá na průběh útoku žádný negativní vliv. Vidíme spuštění serveru s jedním parametrem. Zmíněný parametr je cesta k záznamovému souboru. Pro názornost příkladu je server spuštěn na pozadí pomocí znaku & a zároveň je vypisován obsah záznamového souboru. Příklad vidíme na obrázcích 2.33 a 2.34.

```
root@debian:/mnt/hgfs/sf/src/mms_attack/client# ./client 192.168.88.213 "Test no ARP poison"
connection estabilished
```

Obr. 2.33: Spuštění klienta s IP adresou MMS serveru a heslem.

Pro provedení útoku zvolil autor utilitu `ettercap`. Pomocí zmíněné utility provedl autor ARP (Address Resolution Protocol) poison útok na klienta, kdy se vydával za serverovou stanici. Na klientově straně došlo k přepsání záznamu pro překlad IP adres na MAC adresy a klient odesílal veškerou komunikaci směřující k serveru přes útočníka. Narozdíl od obvyklého způsobu, kdy útočník zaujímá pozici výchozí



Obr. 2.34: Zachycení spojení pravým MMS serverem.

brány, pro zachyceni veškeré komunikace z daného zařízení, nám stačí zachytit komunikaci pouze se serverem, jiná nás nezajímá. Klient má IP adresu 192.168.88.198, server 192.168.88.213 a útočník s podvrženým serverem má IP adresu 192.168.88.208. Parametry pro spuštění nástroje ettercap jsou ve výpisu 2.9. V případě, že máme odlišnou topologii sítě, musíme parametry adekvátně upravit.

Výpis 2.9: Parametry utility `ettercap` pro provedení ARP poison útoku.

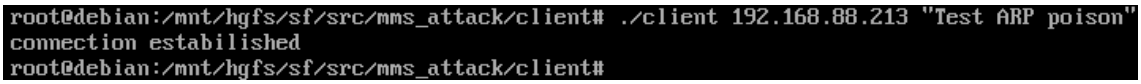
```
ettercap -i eth0 -T -M ARP /192.168.88.198//
  /192.168.88.213//
```

Po úspěšném otrávení ARP tabulek na klientské a serverové stanici útočník přistoupí ke spuštění podvrženého serveru, který je popsán v odstavci výše. Zároveň nesmíme zapomenout zapnout možnost přeposílání IP paketů, viz výpis 2.10. V případě, že tak neučiníme, nebude klient schopen komunikovat se serverem a obsluha klienta může pojmout podezření na probíhající útok.

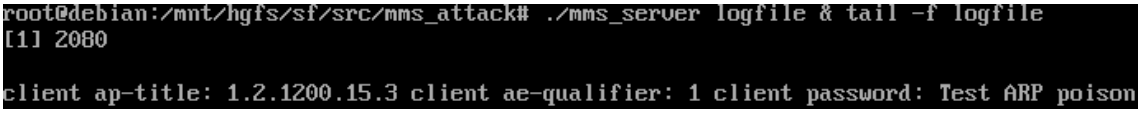
Výpis 2.10: Nastavení přeposílání komunikace mezi klientem a serverem

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Avšak jak vidíme na obrázcích 2.35 a 2.36, klient je stále schopen komunikovat s pravým serverem a nepřipojuje se k serveru útočníka.



Obr. 2.35: Spuštění klienta s IP adresou MMS serveru a heslem po provedení útoku ARP poison.



Obr. 2.36: Zachycení spojení pravým MMS serverem i přes otrávení ARP záznamů.

Pro úspěšné přesměrování komunikace musí útočník na svém stroji nastavit příslušná pravidla pro firewall. Abychom snížili šanci na odhalení, nesmíme přesměrovat veškerou komunikaci mezi klientem a validním serverem, ale pouze komunikaci mezi

klientem a serverem. Obsluha klienta musí být schopna komunikovat se strojem, na kterém běží server, jinak může pojmout podezření o probíhajícím útoku. Protokol MMS spoléhá na zapouzdření do protokolu TCP, přičemž využívá služeb ISO TSAP (Transport Service Access Point) třídy 0. Zmíněná služba v praxi nejčastěji využívá protokol 102, proto budeme přesměrovávat komunikaci právě z tohoto portu klienta na stejný port útočícího stroje. V případě, že služba bude využívat jiný port, není problém jej sledováním komunikace zjistit. Pro modifikaci pravidel pro firewall využijeme nástroj `iptables`. Přesné použití můžeme vidět na výpisu 2.11. Pravidlo nastaví firewall do režimu NAT (Network Address Translation), kdy bude probíhat překlad adres a v případě průchodu paketu obsahující protokol TCP na port 102 jej přesměrujeme pomocí parametru `-j REDIRECT` na lokální adresu útočícího stroje s portem 102. Nesmíme zapomenout zálohovat stávající pravidla firewallu příkazem `iptables-save > savedrules.txt`. Pro jejich obnovení pak využijeme příkaz `iptables-restore < savedrules.txt`.

Výpis 2.11: Nastavení pravidla firewallu pro přesměrování komunikace z portu 102 na port 102 útočícího stroje.

```
iptables -t nat -A PREROUTING -p tcp --destination-port
    102 -j REDIRECT --to-port 102
```

Přidání pravidla ověříme příkazem `iptables -t nat -L -n -v`. Nástroj `iptables` vytiskne tabulku s pravidly. Tabulku můžeme vidět na obrázku 2.37.

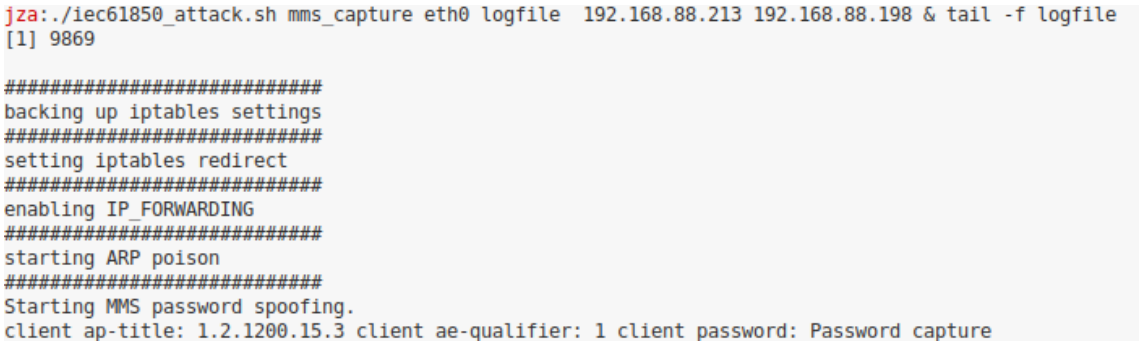
```
[root@jza /home/jza/Documents/skola/diplomka/zdrojaky/src/attack_util]# iptables -t nat -L -n -v
Chain PREROUTING (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 REDIRECT   tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           tcp dpt:102 redir ports 102

Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain POSTROUTING (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

Obr. 2.37: Tabulka pravidel firewallu.

Klient se připojuje stále stejným způsobem na stejnou adresu s heslem "Password capture", viz obrázek 2.38. Avšak jak můžeme vidět na obrázku 2.39, místo validního serveru na adrese 192.168.88.213 se připojil na útočníkův server na adrese 192.168.88.208.

```
root@debian:/mnt/hgfs/sf/src/mms_attack/client# ./client 192.168.88.213 "Password capture"
connection established
```

Obr. 2.38: Spuštění klienta po úspěšném MITM útoku s heslem "Password capture".



Obr. 2.39: Komunikace přesměrována na útočníkovu stanici zachytávající hesla.

Na obrázku 2.39 můžeme vidět spouštění útoku pomocí automatizačního nástroje vytvořeného autorem pro zjednodušení a urychlení práce. Nástroj samotný bude popsán v následující části této práce, konkrétně v sekci 2.5. Krom spouštění útoku můžeme vidět provádění jednotlivých, výše popsaných částí. Zároveň lze na posledním řádku vidět zachycené klientovo heslo.

Po úspěšném provedení útoku musíme zahladit stopy sekvencí příkazů v seznamu 2.1. Jak může čtenář vidět, nejdříve musíme ukončit aplikaci `ettercap`, která zajišťuje otrávení ARP záznamů na klientské a serverové stanici. ARP záznamy na jednotlivých stanicích se postupně obnoví do správného nastavení. Zároveň musíme smazat stopy po modifikaci pravidel firewallu, přičemž využijeme dříve zmíněného příkazu `iptables_restore`. V dalším kroku smažeme soubor savedrules.txt, do kterého jsme si uložili původní konfiguraci firewallu. Nesmíme zapomenout odstranit pravidla pro překlad adres pomocí NAT protokolu čtvrtým příkazem v seznamu 2.1, kde místo proměnné `<číslo pravidla>` dosadíme číslo námi přidaného pravidla pro přesměrování komunikace na portu 102. Jako poslední zakážeme přeposílání IP paketů zapsáním nuly do příslušného souboru pomocí posledního příkazu v seznamu 2.1.

Seznam 2.1: Posloupnost příkazů pro zahlazení stop na útočícím stroji.

- `kill $(ps aux | grep ettercap | awk 'print $2')`
- `iptables-restore < savedrules.txt`
- `rm savedrules.txt`
- `iptables -t nat -D PREROUTING <číslo pravidla>`
- `echo 0 > /proc/sys/net/ipv4/ip_forward`

2.5 Testovací nástroje pro usnadnění práce a konfigurace útoků

V následujícím textu budou čtenáři přiblíženy autorem vytvořené knihovny a nástroje pro automatizaci a konfiguraci výše popsaných útoků. Vzhledem k nedostatku volně přístupných nástrojů použitelných pro tuto práci autor vytvořil podpůrné knihovny, konkrétně parser protokolu GOOSE s jednoduchým parserem konfiguračních souborů a automatizační skript pro spouštění dílčích útoků. Návrh, implementaci a použití si vysvětlíme v částech, které se blíže věnují každému nástroji.

2.5.1 Parser konfiguračních souborů

Při prvních pokusech o provádění útoků autor narazil na problém zdlouhavého měnění konfigurací programů implementujících tyto útoky. Každá změna konfigurace vyžadovala zásah do zdrojových kódů programů, jejich následný překlad a linkování a až poté spuštění. Autorovi přišel zmíněný přístup jako příliš časově náročný, a tedy nevhodný pro vývoj. Proto se rozhodl napsat jednoduchý, přenosný a univerzální parser konfiguračních souborů.

Parser nesměl být použitelný pouze pro jeden projekt, v našem případě útok. Vzhledem k rozdílům možných konfigurací mezi jednotlivými útoky nesměl být parser napsán podle modelu, kdy každá položka v konfiguračním souboru má svoji vlastní metodu, která přečte a vrátí její hodnotu. Vzhledem ke specializaci řešení nebylo možné jej využít pro všechny útoky a software, který bude vyvíjen v budoucnu, jelikož by každý projekt vyžadoval svou implementaci modulu.

Jazyk C++, ve kterém jsou všechny útoky zmíněny v práci implementovány, od roku 2003, kdy vyšel standard C++03, podporoval knihovnu STL (Standard Template Library) [21]. Vznikl model šablonového metaprogramování, kdy programátor napsal obecnou šablonu a překladač během překladu provedl specializaci pro daný datový typ. Autor se zmíněným modelem inspiroval a využil možnosti, které nabízí pro vývoj tohoto parseru konfiguračních souborů.

Modul je implementován v jediném hlavičkovém souboru `config_parser.h`, který obsahuje třídu `config_parser`. Modul využívá pouze standardní knihovny jazyka C++. Použité knihovny jsou: `iostream` pro definici základních datových typů a řetězců, `fstream` pro práci se soubory, `sstream` pro práci s řetězci, konkrétně konverzi obecných datových typů na návratový typ šablony popsané níže, `vector` pro tokenizaci vstupních řetězců a `algorithm` pro odstranění bílých znaků v řetězcích. Volba standardních knihoven umožňuje importování modulu do projektů bez nutnosti instalovat dodatečné knihovny, stačí pouze zahrnout hlavičkový soubor do projektu pomocí makra preprocesoru `#include <config_parser.h>`. Pro

vysvětlení významu jednotlivých položek podporuje parser řádkové komentáře.

Implementace V seznamu 2.2 můžeme vidět privátní členy třídy. Obě metody, které můžeme vidět v seznamu níže, jsou statické s nastaveným příznakem `inline` pro lepší optimalizaci překladačem. Symbol & před názvy parametrů znamená, že parametry jsou předávány pomocí reference, tedy odkazu, z důvodu ušetření zbytečného kopírování paměti rozsáhlých objektů. Klíčové slovo `const` před datovým typem parametru zajišťuje neměnnost předaných dat v těle metody. První metoda vrací binární hodnotu nabývající hodnot pravda či nepravda a určuje, jestli je daný řetězec komentář. Druhá metoda vrací kopii řetězce, který je prvním členem vektoru tokenů předaných odkazem jako parametr metody. Řetězec `path` je naplněn inicializačním seznamem při instanciaci objektu parseru.

Seznam 2.2: Privátní členy třídy `config_parser`.

- `bool is_comment(const std::string &comment)`
- `std::string get_first(const std::vector<std::string> &tokens)`
- `std::string path`

Veřejné rozhraní třídy tvoří, jak lze vidět na seznamu 2.3, tři metody. Konstruktor, šablona `get_item` a statická metoda `tokenise`. Konstruktor s šablonou pro čtení konfiguračních souborů mají parametry konstantní reference na řetězec, důvody jsme si vysvětlili výše. V případě konstruktoru cestu ke konfiguračnímu souboru a v případě šablony `get_item` název položky konfiguračního souboru, jejíž hodnotu chceme získat. Autor zvolil šablonu z důvodu jednoduchosti použití a obecnosti, kterou šablony umožňují. Použití si vysvětlíme v následujícím odstavci. Metoda `tokenise` má první parametr konstantní referenci na vstupní řetězec, který chceme rozdělit na tokeny, druhý parametr je dělící znak, podle kterého předaný řetězec dělíme.

Seznam 2.3: Veřejné členy třídy `config_parser`.

- `config_parser(const std::string &input)`
- `template<typename TYPE> TYPE get_item(const std::string &input)`
- `std::vector<std::string> tokenise(const std::string &line, char delim)`

Použití Pro použití parseru stačí s pomocí makra `#include <config_parser.h>` vložit knihovnu. Následně vytvoříme instanci zmíněné třídy, je na uživateli, zda

na zásobníku nebo na haldě, avšak autor v případě využití haldy doporučuje použít technologii chytrých ukazatelů. Konstruktoru dané instance dáme jako argument cestu ke konfiguračnímu souboru a následně zavoláme šablonu `get_item` se specifikovaným návratovým datovým typem. Příklad použití na konfiguračním souboru 2.12 lze vidět níže ve výpisu 2.13.

Výpis 2.12: Příklad konfiguračního souboru.

```
VALID_BOOL = true # this is a comment
# this is another comment
VALID_STRING = 192.168.88.253 # this is comment as well
VALID_INT = 42 # answer to life, the universe and
   everything
VALID_DOUBLE = 750.5.5 # WATCH OUT! Parser will read
   750.5
```

Výpis 2.13: Příklad volání šablony `get_item`.

```
config_parser parser(path);
bool valid_bool = parser.get_item<bool>("VALID_BOOL");
std::string valid_string = parser.get_item<std::string
   >("VALID_STRING");
int valid_int = parser.get_item<int>("VALID_INT");
double valid_double = parser.get_item<double>("
   VALID_DOUBLE");
```

2.5.2 Parser protokolu GOOSE

Pro útok GOOSE replay potřeboval autor nástroj pro čtení záznamů síťového provozu, konkrétně protokolu GOOSE. Jediná použitelná knihovna se zdála být knihovna napsaná docentem Dušanem Kolářem z Fakulty Informačních Technologií Vysokého Učení Technického v Brně [22]. Knihovna je napsána v jazyce C pro co nejmenší zátěž systému. Bohužel pro účely této práce je nevhodná z několika důvodů. První důvod je Open Source Licence VUT v Brně , pod kterou je knihovna zveřejněna. Licence vyžaduje šíření díla odvozeného z knihovny v otevřené formě, tedy ke každé binární distribuci musí být i zdrojové kódy programů používající danou knihovnu. Pro případné navázání na tuto práci může dojít k potřebě šířit programy používající knihovnu GOOSE parser v uzavřené formě, přičemž licence zmíněná výše bude klást významnou překážku. Druhým důvodem pro nevyužití dané knihovny je její obecná povaha, kdy jí chyběla funkcionalita potřebná k provedení zvoleného útoku. Autor je schopen a mohl potřebnou funkcionalitu doplnit, avšak použití a šíření knihovny

bude opět omezeno z předchozího důvodu, jelikož Open Source Licence VUT v Brně se vztahuje i na díla odvozená. Z výše zmíněných důvodů se autor rozhodl napsat svou vlastní knihovnu pro parsování protokolu GOOSE. Vzhledem ke znalostem získaných během studia pro napsání této práce napsání a odladění knihovny nezabralo mnoho času.

Implementace Narozdíl od předchozího nástroje si kvůli rozsahu knihovny nemůžeme dovolit stejně podrobný popis, proto se omezíme pouze na základní informace. Knihovna je, stejně jako většina programového vybavení vytvořeného pro tuto práci, napsána v jazyce C++. Pro zajištění kompatibility napříč širokou škálu systémů využívá knihovna pouze standardní knihovny jazyka C a C++. Konkrétně `iostream`, `cstring` a `netinet/in.h`. Knihovna je tvořena pouze jedinou třídou obsahující celou implementaci. Díky faktu, že protokol GOOSE je tvořen položkami typu TLV (Type, Length, Value), kdy první konstanta identifikuje typ dat, druhá konstanta určuje délku dat v bytech a poslední část je sekvence bytů obsahující daná data. Vzhledem k rozporu identifikačních konstant v materiálech k protokolu GOOSE, nacházejících se v [17], použil autor konstanty použité v komunikaci ve virtuální infrastruktuře, které získal pomocí nástroje Wireshark. Identifikační konstanty můžeme vidět v tabulce 2.2.

Tab. 2.2: Konstanty identifikující typ položky protokolu GOOSE.

Název konstanty	Hodnota v šestnáctkové soustavě
GOOSE_CBREF	0x80
GOOSE_TTL	0x81
GOOSE_DATASET	0x82
GOOSE_GOID	0x83
GOOSE_TIME	0x84
GOOSE_STNUM	0x85
GOOSE_SQNUM	0x86
GOOSE_TEST	0x87
GOOSE_CONVREV	0x88
GOOSE_NDSCOM	0x89
GOOSE_NUMBER	0x8a
GOOSE_DATA	0xab
GOOSE_DATA_INTEGER	0x85
GOOSE_DATA_FLOATING	0x87
GOOSE_DATA_UTC_TIME	0x91

Pro přesnější popis jednotlivých metod se můžeme podívat do dokumentačních komentářů systému Doxygen v zdrojových souborech implementační částí diplomové práce.

Použití Parser zahrneme do projektu voláním makra `#include <goose_parser.h>`. Pro vytvoření objektu `goose_data` obsahující implementaci parseru předáme konstruktoru odkaz na první byte paketu protokolu GOOSE. Parser následně čte a ukládá jednotlivé hodnoty do svých privátních proměnných. Hodnotu privátních proměnných může uživatel získat voláním veřejných metod obecně označovaných jako getter. Informace o jednotlivých getterech jsou popsány ve zdrojových kódech programu. Po přečtení datové části paketu musí uživatel ručně procházet data pomocí veřejných metod, přičemž ukazatel na první byte datové části paketu získá voláním `getData()` a velikost datové části získá voláním `getDataSize()`. Nejdříve musí získat délku TLV pomocí volání metody `tlv(data_pointer, type, data_size)`, kde `data_pointer` obsahuje ukazatel na začátek dat, které chceme číst, `type` je reference na celočíselnou hodnotu označující typ dat a kam bude daná hodnota uložena a `data_size` je reference na velikost zbývajících dat. Následně uživatel využije konstrukci `switch` pro větvení programu na základě jednotlivých typů dat. Po přečtení položek se uživatel musí posunout v paketu voláním `move_in_packet(data_pointer, data_size, tlv_len)`, kde `data_pointer` je ukazatel na začátek jednotky TLV, `data_size` je reference na velikost zbývajících dat a `tlv_len` je délka dané TLV jednotky.

2.5.3 Automatizační nástroj

Pro zrychlení vývoje s testováním, sjednocení rozhraní a umožnění automatizovaného testování zařazením jednotlivých útoků do větších programových celků vytvořil autor automatizační nástroj. Nástroj je napsán ve skriptovacím jazyce Bash a je vytvořen pro příkazovou řádku operačního systému Linux distribuce Kali.

Návrh a implementace Jak již autor zmínil, nástroj je napsán pro operační systém Kali Linux. Daná distribuce byla zvolena z důvodu již předinstalovaného programového vybavení pro jednodušší vedení síťových útoků. Nástroj může být použit i na jiných distribucích operačního systému Linux, ale v tom případě je nutno doinstalovat aplikace, na kterých je tento nástroj závislý. Konkrétní aplikace jsou `macchanger` a `ettercap`.

Autorem napsaná aplikace podporuje útoky GOOSE flood, GOOSE replay, GOOSE stNum, GOOSE semantic a MMS password capture, přičemž spouštění jednotlivých útoků je pro jednoduchost, udržovatelnost a rozšiřitelnost nástroje napsáno

pomocí funkcí, což umožňuje jednoduchou škálovatelnost. Každý útok obnáší manipulaci se síťovým rozhraním, takže je nutno nástroj spouštět jako uživatel `root`. V případě útoků na protokol GOOSE dochází před zahájením útoku k vypnutí zvoleného rozhraní, změnění MAC adresy síťového adaptéru na MAC adresu zadanou jako jeden z argumentů při spuštění nástroje a opětovnému spuštění síťového rozhraní. Po změně MAC adresy na uživatelem zvolenou MAC adresu dojde ke spuštění samotného útoku. Útok vypneme kombinací kláves `Ctrl+C`. Dojde k vypnutí útočící aplikace, vypnutí síťového rozhraní, změně MAC adresy adaptéru na původní MAC adresu a opětovnému spuštění síťového rozhraní.

V případě volby útoku MMS password capture dojde k záloze pravidel firewallu, přidání pravidla pro přesměrování komunikace z portu 102 na náš port 102, zapnutí nástroje `ettercap` pro nutný útok ARP poison, povolení přeposílání IP paketů skrz naše zařízení a poté dojde ke spuštění samotného podvrženého MMS serveru pro zachytávání hesel, který ukládá hesla do záznamového souboru, jehož cesta je jedním z argumentů spouštění automatizačního nástroje. Po ukončení útoku kombinací kláves `Ctrl+C` dojde k ukončení serveru, vypnutí aplikace `ettercap` pomocí PID (Process ID), které jsme si zaznamenali při spuštění aplikace. Dále dojde k zakázání přeposílání paketů IP skrz naše zařízení, obnovení pravidel firewallu a odstranění pravidla pro přesměrování komunikace na portu 102 na port 102 našeho zařízení. Na závěr se smaže záloha pravidel pro firewall.

Použití Důležité je udržet danou adresářovou strukturu, kdy jsou adresáře útoků a adresář s tímto nástrojem na stejné úrovni. Před zahájení útoků je třeba přeložit veškeré zdrojové soubory. Překlad se provede otevřením adresáře `src` v odevzdaných souborech a zavoláním programu `make`. `Make` nejdříve přeloží veškeré potřebné knihovny, které se v daném repozitáři nacházejí a pak přejde k překladu dílčích útoků. Pokud uživatel používá verzovací nástroj Git, musí repozitář klonovat s příznakem `--recursive` pro stažení knihoven, které jsou importovány jako jednotlivé submoduly. V případě, že jsou složky `libiec61850` a `libtins` prázdné, stačí knihovnu stáhnout ve formě zdrojových souborů a zkopírovat do příslušných složek.

V případě virtualizovaného prostředí je třeba si dávat pozor na volání jednotlivých útoků, aby rozhraní používané pro vedení útoku GOOSE replay, GOOSE stNum a GOOSE semantic bylo jiné než rozhraní, které používají virtuální stroje. Z důvodu vyžadovaného čtení ze sítě dochází při volbě stejných rozhraní ke kolizi a nedobrovolnému zamrznutí počítače. V případě útoků na protokol GOOSE se používané síťové rozhraní nastavuje v konfiguračním souboru pro daný útok. Pokud by měl čtenář nejasnosti, autor čtenáře odkazuje na rozsáhlou nápovědu automatizačního nástroje, kterou lze vyvolat zadáním argumentu `--help` či jeho krátké formy `-h`.

Dílčí příklady použití můžeme vidět ve výpisu 2.14.

Výpis 2.14: Příklady spouštění autorem napsaného nástroje.

```
./iec61850_attack.sh flood ../goose_flood/config.cfg
   00:0c:29:3c:06:3d
./iec61850_attack.sh stnum ../goose_stnum/config.cfg
   00:0c:29:3c:06:3d
./iec61850_attack.sh semantic ../goose_semantic/config.
   cfg 00:0c:29:3c:06:3d
./iec61850_attack.sh replay ../goose_replay/config.cfg
   ../goose_replay/goose_replay_recording.pcapng 00:0c
   :29:3c:06:3d
./iec61850_attack.sh mms_capture eth0 logfile
   192.168.88.213 192.168.88.198
```

2.6 Návrh mitigačních opatření

Vzhledem k nekompletnosti řešení poskytnutým standardem IEC 62351, v našem případě částmi 4 a 6, neexistuje jednotná verze mitigačních opatření pro protokoly standardu IEC 61850. V následující sekci autor popisuje jednotlivá mitigační opatření pro každý implementovaný útok. Jejich výhody, nevýhody, a důvody proč se rozhodl pro taková opatření, která zde zmiňuje.

Pro zajištění bezpečné komunikace potřebujeme zajistit důvěrnost zpráv, integritu zpráv, autenticitu a nepopiratelnost. Zmíněné vlastnosti zabraňují: čtení odesílaných zpráv, modifikaci zpráv během přenosu mezi zařízeními, podvržení identity zařízení a podvržení události samotné.

Jedním z mechanismů pro zajištění bezpečné komunikace je šifrování. Tvrzení je pravdivé pouze v ideálním případě, kdy nemáme časová zpoždění způsobena konstruováním paketu, jeho šifrováním, odesláním, přenosem, přijetím a zpracováním. V reálné aplikaci definuje standard pro aplikace pracující v reálném čase nutnost doručení zprávy do 3 milisekund po jejím vzniku. V následujících odstavcích se blíže podíváme na implementace jednotlivých symetrických i asymetrických šifer a zaměříme se na jejich časovou náročnost.

RSA Nejdříve se podíváme na šifrovací algoritmus RSA. Jedná se asymetrickou šifru založenou na kryptografii veřejných klíčů. Algoritmus lze využít jak pro šifrování komunikace, což nám zajistí důvěrnost přenášených dat, tak i jako formu elektronického podpisu, což nám zajistí autenticitu odesílatele. Pro vytvoření elektronického podpisu je paket podepsán privátním klíčem odesílatele a podpis je ověřen

na straně příjemce veřejným klíčem odesílatele. Když chceme využít šifru pro zašifrování dat, odesílatel zašifruje zprávu veřejným klíčem příjemce, který použije svůj privátní klíč k dešifrování zprávy.

Generování klíčového páru můžeme vidět v algoritmu 1. Mějme vstupy, prvočísla p_1, p_2 a výstupy, klíče vyžadované algoritmem RSA, e a d. Pro vygenerování klíčů zvolíme dvě stejně velká prvočísla, ideálně alespoň 2048 bitová. Vypočítáme modulus n a hodnotu Eulerovy funkce pro zvolená p_1 a p_2. Následně zvolíme kladné celé číslo tak, že platí $1 < e < \phi(n)$ a je nesoudělné s výsledkem Eulerovy funkce. Zde je e nazýván veřejný klíč. Privátní klíč d je převrácenou hodnotou veřejného klíče e.

Algoritmus 1: Generování klíčového páru RSA.

Vstup: p_1, p_2
Výstup: e, d
1: $n = p_1 * p_2$
2: $\phi(n) = (p_1 - 1)(p_2 - 1)$
3: $1 < e < \phi(n), NSD(e, \phi(n)) = 1$
4: $e.d \mod \phi(n) = 1$

Pro zašifrování zprávy nejdříve musíme získat veřejný klíč (e, n) příjemce. Po získání klíče zašifrujeme zprávu m v otevřeném formátu pomocí vztahu $c = m^e \mod (n)$ a šifrovanou zprávu odešleme příjemci. Příjemce provede dešifrování zprávy pomocí svého privátního klíče (d, n) a operace $m = c^d \mod (n)$ [23].

Ačkoliv je daný algoritmus s délkou klíče 2048 bitů považován za bezpečný, šifrování a dešifrování zpráv trvá moc dlouho pro splnění těsných časových termínu. Autor proto provedl testy jednotlivých šifer implementovaných pomocí knihovny Crypto++. Výkonnostní test verze 5.6.0 zjistil, že zašifrování dat 2048 bitovým klíčem trvá 0.16 milisekundy a následné dešifrování trvá 6.08 milisekund. Elektronický podpis klíčem stejné délky trvá 6.05 milisekund a jeho ověření trvá 0.16 milisekund [24]. Autor provedl opětovné testování verze 8.2 na počítači s procesorem AMD Ryzen 5 1600, taktovaným na 3.2 GHz. Výkonnostní testování ukázalo, že zašifrování dat 2048 bitovým klíčem trvá 0.024 milisekund a dešifrování trvá 1.139 milisekund. Podepsání dat trvalo 1.141 milisekund a jeho ověření trvalo 0.025 milisekund.

Jak vidíme v předchozím odstavci, kryptografické nároky pro zašifrování a podepsání zprávy tvoří 2.329 milisekund, přičemž nároky na šifrování a dešifrování jsou 1.163 milisekund a nároky na podepsání a ověření podpisu jsou 1.166 milisekund. V případě využití obou metod nezbývá mnoho času pro konstrukci paketu, jeho odeslání a následné zpracování. Test byl proveden na procesoru pro stolní počítač, přičemž procesory používané v průmyslových zařízeních nedosahují takových výkonů. Uživatel se proto musí zamyslet, zda je pro něj důležitější zajištění důvěrnosti, nebo autenticity spolu s nepopiratelností, přičemž nedojde k zajištění integrity zpráv.

Změna může nastat v případě vývoje hardwarové implementace daných algoritmů navržených například v [23].

HMAC Pro zajištění integrity i autenticity odesílaných dat lze využít technologii HMAC (Hash-based Message Authentication Code), které již v roce 2003 potřebovaly cca. 10 mikrosekund na vygenerování HMAC pro typický IP paket, dokonce i při použití softwarové verze bylo třeba pouze cca. 30 mikrosekund [18].

Technologie HMAC pracuje na principu kryptografických hashovacích funkcí s privátními klíči pro zajištění integrity a autenticity dat. Princip lze vidět v algoritmu 2. Mějme vstupy k, m, *opad*, *ipad* a kryptografickou hashovací funkci H, například SHA-256 (Secure Hash Algorithm) [25]. Operátor $\oplus$ značí bitovou operaci XOR, operátor $||$ provádí konkatenaci dat, k je soukromý klíč integrovaný v zařízení, *opad* a *ipad* jsou dvě konstanty 0x5C pro *opad*, respektive 0x36 pro *ipad* o velikosti základního bloku.

Algoritmus 2: Vytvoření podpisu pomocí technologie HMAC.

Vstup: k, m, *opad*, *ipad*

Výstup: p

$HMAC(k, m) = H(k \oplus opad||H(k \oplus ipad||m))$

Autoři článku [25] navrhli aplikaci pro programovatelné hradlové pole, která je schopna pracovat na frekvenci 116.24 MHz v režimu bez maskování spotřeby, nebo na frekvenci 87.08 MHz v režimu s maskováním spotřeby. V případě režimu maskování spotřeby je aplikace odolná vůči útoku postranním kanálem využívajícím proměnlivou spotřeby energie v závislosti na vstupních datech. V tabulce 2.3 můžeme vidět srovnání výstupních rychlostí v jednotlivých režimech, přičemž s režimem pro maskování spotřeby je výstupní rychlost 655.66 $\mathrm{Mbit\,s^{-1}}$ a bez režimu maskování spotřeby je výstupní rychlost 875.22 $\mathrm{Mbit\,s^{-1}}$ [25].

Režim	Maximální frekvence	Výstupní rychlost
Bez maskování spotřeby	116.24 MHz	875.22 $\mathrm{Mbit\,s^{-1}}$
S maskováním spotřeby	87.08 MHz	655.66 $\mathrm{Mbit\,s^{-1}}$

Tab. 2.3: Srovnání výkonu aplikace pro generování HMAC s a bez režimu pro maskování spotřeby.

Údaje v tabulce výše říkají, že jsme schopni podepsat, a tedy ověřit integritu 109 402 500 bytů za sekundu. Za milisekundu jsme schopni podepsat 109 402 bytů. Nesmíme zapomenout, že ověření trvá stejně dlouho jako podepsání, takže musíme zdvojnásobit potřebný čas. Abychom dosáhli času 2 milisekundy na konstrukci, podepsání, odeslání, přenos, zachycení, zpracování a ověření, můžeme si dovolit vymezit pro algoritmus HMAC maximálně 0.5 milisekundy. Budeme tedy schopni pode-

psat 54 701 bytů. Při velikosti paketu protokolu GOOSE nepřekračující 850 B, jsme schopni zajistit integritu celého přenášeného paketu za předpokladu, že neunikly privátní klíče používané k podpisu a následnému ověření. V případě úniku klíčů je třeba okamžitě přejít na novou sadu klíčů, jinak nejsme schopni ověřit integritu ani autenticitu zpráv.

AES Další možností je využití blokové šifry AES (Advanced Encryption Standard) s klíčem o délce 128 bitů. Díky implementaci pro programovatelné pole hradel je schopna šifra AES zajistit průchodnost až 37.21 Gbit s^{-1} [26]. Při této průchodnosti jsme schopni zajistit šifrování paketů protokolu GOOSE pro jakoukoliv v praxi používanou velikost. V případě využití šifry AES autor navrhuje pro vytvoření a sdílení klíče využít libovolné varianty Diffie-Hellman algoritmu. Algoritmus Diffie-Hellman je obecně známý. Můžeme si jej připomenout níže ve výpisu algoritmu 3.

Algoritmus 3: Základní implementace algoritmu Diffie-Hellman.

Vstup: p, g, a, b

Výstup: key

1: Alice s Bobem se skrz nezabezpečený kanál shodnou na použití modulu p a základu g.

2: Alice zvolí tajné celé číslo a, a pošle Bobovi A, kde $A = g^a \mod p$.

3: Bob zvolí tajné celé číslo b, a pošle Alici B, kde $B = g^b \mod p$.

4: Alice vypočítá $key = B^a \mod p$.

5: Bob vypočítá $key = A^b \mod p$.

V případě, že máme dostatečně dlouhá prvočísla p a q, je náročnost výpočtu problému diskrétního logaritmu dostatečně náročná pro zajištění utajení sdíleného klíče. Pro mitigaci útoku Man in the Middle je třeba zvolená p a q podepisovat důvěryhodnou certifikační autoritou, což zabrání podvržení výchozích hodnot útočníkem a výslednému získání samotného sdíleného klíče. Zmíněnou certifikační autoritu je třeba udržovat v aktuálním stavu softwarového vybavení, hlavně pomocí aktualizací bezpečnostních mechanismů, jelikož její kompromitování jako klíčového ověřovatele zvolených výchozích hodnot pro generování sdíleného klíče zcela znehodnotí funkcionalitu navrhovaného opatření. V případě, že bude útočník schopen zjistit sdílený klíč, bude schopen dešifrovat komunikaci obou zařízení. Pro tento případ je nutno používané klíče pravidelně obnovovat spolu s nově navázaným spojením. Při zvolení klíče dostatečné bitové délky se zdá býti týdenní obnova dostatečná. V případě odposlechu síťového provozu útočníkem bude zjištění klíče a spojení náročné, jelikož přesný moment vytvoření spojení s novým sdíleným klíčem bude ztracen v kvantech dat produkovaných zařízeními v celé síti.

TLS V předchozích odstavcích jsme si popsali mitigační opatření směřována na mitigaci útoků na protokoly GOOSE fungující na druhé vrstvě referenčního modelu ISO/OSI. Nyní se zaměříme na mitigační opatření vůči autorem implementovanému útoku Man in the Middle pomocí otrávení tabulky s ARP záznamy na komunikaci protokolem MMS.

Jedním z mechanismů pro vytvoření šifrovaného spojení mezi dvěma stanicemi je využití služeb TLS. Jedná se o službu sloužící k zabezpečení komunikace aplikačních protokolů jako HTTP, SSH a dalších. Patří mezi ně i již zmiňovaný protokol MMS. Samotný protokol MMS pro komunikaci nevyužívá žádné bezpečnostní mechanismy pro zajištění důvěrnosti, autenticity, integrity a pro zajištění zmíněných vlastností spoléhá na protokoly, do kterých je zapouzdřen. Proto můžeme pro zajištění výše zmíněných vlastností využít službu TLS.

Technologie TLS podporuje různé metody navázání spojení, výměny klíčů a následného šifrování, zmíníme jen metody používané TLS verzi 1.3. TLS 1.3 pro navázání spojení používá tři kombinace: DHE-RSA (Diffie-Hellman Exchange - Rivest-Shamir-Adleman), ECDHE-RSA (Elliptic-curve Diffie-Hellman - RSA) a ECDHE-ECDSA (Elliptic Curve Digital Signature Algorithm). Každý ze zmíněných postupů, nejenže zajišťuje bezpečné předání klíče, ale také zajišťuje vlastnost dopředné bezpečnosti, kdy kompromitace klíče neohrozí minulou komunikaci. Následně dochází k šifrování například šifrou AES. Samotné šifrování ale neznamená zajištění autenticity odesílatele. Pro úspěšné ověření potřebujeme zajistit správné podepisování a ověřování certifikátů. Z toho důvodu vyžaduje nasazení TLS vlastní certifikační autoritu. Stejně jako v minulém případě, je třeba udržovat certifikační autoritu aktualizovanou a řádně zabezpečenou, v opačném případě je komunikace zranitelná vůči útokům Man in the Middle.

Další opatření Nejlepší opatření proti útoku na OT síť je neumožnit útočníkovi proniknout do IT sítě, ze které pak může útočit na OT síť. Většina průniků vedoucí k útokům na OT síť je způsobena chybou zaměstnance, který buď schválně, či nedopatřením umožní útočníkovi proniknout do firemní sítě daného průmyslového zařízení. Z toho důvodu je třeba klást důraz na několik věcí. První a nejdůležitější je výběr spolehlivých zaměstnanců a jejich následné proškolení v oboru zásad bezpečnosti. Jak poznat phishingový útok, podvržené stránky, podvržené formuláře pro zachytávání hesel apod. Dalším krokem je správně nastavit přístupová práva nejen k fyzickým strojům, důležité je povolit přístup ke klíčovým prvkům infrastruktury pouze zkušeným a ověřeným pracovníkům, kteří ví, co dělají. Spolu se správným nastavením přístupu je třeba správně navrhnout jak IT síť, tak OT síť, včetně správného nasazení firewallů a mechanismů detekující průniky. V případě detekovaného průniku je třeba dané zařízení izolovat a provést důkladnou analýzu proč se tak stalo.

Vhodné se jeví také nasazení technologie SDN (Software Defined Networking), která umožňuje jednodušší správu a řízení firemní sítě.

Závěr

V rámci této diplomové práce jsme si vysvětlili, co to je kritická infrastruktura, co znamená kritická infrastruktura pro energetický průmysl a jaké následky může mít její poškození. Vysvětlili jsme si, co to je ICS KillChain, jaké má fáze, včetně podrobnějšího dělení a následně jsme si popsali několik virů, které implementují výše zmíněný princip. Zároveň jsme si na praktických příkladech z posledních let ukázali, jak probíhají útoky na průmyslová zařízení, jaké způsobují škody a jaké mají následky.

V praktické části se čtenář dozvěděl vybrané informace o standardech, kterými se autor v této práci zabýval. Můžeme jmenovat IEC 61850 a IEC 62351, přičemž nás zajímá hlavně implementace standardu IEC 61850 pomocí protokolu GOOSE a MMS. V rámci standardu IEC 62351 autor vybral některé podstatné části pro tuto diplomovou práci.

S pomocí vedoucího Ing. Blažka autor analyzoval zvolené vektory útoku, nastudoval potřebné podklady pro jejich provedení, z nichž můžeme jmenovat například knihovny programovacího jazyka C++ `libIEC61850` a `libtin`, přičemž provedl nutné modifikace zmíněných knihoven pro manipulaci s obsahem hlaviček a datových části paketů.

Autor implementoval sedm útoků, přičemž dva útoky byly implementovány pro fyzickou infrastrukturu a zbývajících pět bylo implementováno pro virtualizovanou infrastrukturu poskytnutou vedoucím Ing. Blažkem. Útok na simulační bit protokolu GOOSE s útokem NTP spoofing byly realizovány na fyzické infrastruktuře, a útoky GOOSE replay, GOOSE stNum, GOOSE semantic, GOOSE flood a MMS password capture byly implementovány na virtualizované infrastruktuře. Útoky na protokoly GOOSE a MMS byly úspěšné, útok na protokol NTP byl z důvodů popsaných v dané sekci pouze částečně úspěšný.

Pro zmíněné útoky prováděné na virtualizované infrastruktuře autor implementoval podpůrné nástroje pro zjednodušení vývoje, testování a provádění jednotlivých útoků. Konkrétně se jedná o parser protokolu GOOSE pro provedení útoku GOOSE replay, parser konfiguračních souborů pro konfiguraci dílčích útoků na protokol GOOSE a nástroj pro jednoduché provádění dílčích útoků, který zároveň zachovává možnost konfigurace jednotlivých útoků. Takový nástroj může být posléze použit jako součást rozsáhlejších nástrojů pro penetrační testování. Parser konfiguračních souborů má díky využití šablon vysokou variabilitu použití. Parser protokolu GOOSE má využití například v podobě analyzování záznamů síťového provozu.

Diplomovou práci lze použít jako součást komplexního nástroje pro penetrační testování průmyslových sítí, případně implementovat navržená mitigačních opatření.

Literatura

[1] MVČR, "Ochrana kritické infrastruktury," [online], 2019. [Online]. Available: https://www.mvcr.cz/cthh/clanek/ochrana-kriticke-infrastruktury-ochrana-kriticke-infrastruktury.aspx

[2] ČEPS, [online], 2019. [Online]. Available: https://ceps.cz/

[3] "Final report on the august 14, 2003 blackout in the united states and canada: Causes and recommendations," U.S.-Canada Power System Outage Task Force, Tech. Rep., duben 2004. [Online]. Available: https://www.energy.gov/sites/prod/files/oeprod/DocumentsandMedia/BlackoutFinal-Web.pdf

[4] M. of North America blackout 2003, [online], 2019. [Online]. Available: https://commons.wikimedia.org/w/index.php?curid=3201241

[5] M. J. Assante and R. M. Lee, "The industrial control systemcyber kill chain," SANS Institute, Tech. Rep., říjen 2015. [Online]. Available: https://www.sans.org/reading-room/whitepapers/ICS/industrial-control-system-cyber-kill-chain-36297

[6] J. Nazario, "Blackenergy ddos bot analysis," Arbor Networks, Tech. Rep., 2007. [Online]. Available: http://atlas-public.ec2.arbor.net/docs/BlackEnergy+DDoS+Bot+Analysis.pdf

[7] ThreatStop, "Black energy," [online], únor 2016. [Online]. Available: https://www.threatstop.com/sites/default/files/threatstop_blackenergy.pdf

[8] A. Cherepanov and R. Lipovsky, "Blackenergy – what we really know about the notorious cyber attacks," ESET, Tech. Rep., říjen 2016. [Online]. Available: https://www.virusbulletin.com/uploads/pdf/magazine/2016/VB2016-Cherepanov-Lipovsky.pdf

[9] Secureworks, "Blackenergy version 2 threat analysis," [online], březen 2010. [Online]. Available: https://www.secureworks.com/research/blackenergy2

[10] Microsoft, [online], 2019. [Online]. Available: https://docs.microsoft.com/en-us/security-updates/securitybulletins/2008/ms08-025

[11] A. Cherepanov, "Win32/industroyer a new threat for industrial control systems," ESET, Tech. Rep.

[12] N. Falliere, L. O. Murchu, and E. Chien, "W32.stuxnet dossier - version 1.4," Symantec Security Response, Tech. Rep., únor 2011. [Online]. Available: https://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/w32_stu%xnet_dossier.pdf

[13] S. Karnouskos, "Stuxnet worm impact on industrialcyber-physical system security," [online], listopad 2011.

[14] A. Matrosov, E. Rodionov, D. Harley, and J. Malcho, "Stuxnet under the microscope," ESET, Tech. Rep.

[15] R. M. Lee, M. J. Assante, and T. Conway, "German steel mill cyber attack," SANS Industrial Control Systems, Tech. Rep., prosinec 2014. [Online]. Available: https://ics.sans.org/media/ICS-CPPE-case-Study-2-German-Steelworks_Facility.pdf

[16] ——, "Analysis of the cyber attack on the ukrainian power grid," SANS Industrial Control Systems, Tech. Rep., březen 2016. [Online]. Available: https://ics.sans.org/media/E-ISAC_SANS_Ukraine_DUC_5.pdf

[17] P. Matoušek, "Description of iec 61850 communication," Faculty of Information Technology, Brno University of Technolgy, Božetěchova 2, Brno, Tech. Rep. FIT-TR-2018-1, září 2018. [Online]. Available: https://www.fit.vut.cz/research/publication-file/11832/TR-61850.pdf

[18] R. Schlegel, S. Obermeier, and J. Schneider, "A security evaluation of iec 62351," *Journal of Information Security and Applications*, vol. 34, pp. 197 – 204, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2214212616300771

[19] J. Hudec, "Generátor útoků na scada protokoly. bakalářská práce," 2019.

[20] E. B. A. Malhotra, I. E. Cohen and S. Goldberg, [online], 2015. [Online]. Available: https://eprint.iacr.org/2015/1020.pdf

[21] cplusplus.com, [online], 2020. [Online]. Available: http://www.cplusplus.com/info/history/

[22] D. Kolář, [online], 2018. [Online]. Available: https://www.fit.vut.cz/research/product/590/.cs

[23] S. D. Thabah, M. Sonowal, R. U. Ahmed, and P. Saha, "Fast and area efficient implementation of rsa algorithm," *Procedia Computer Science*, vol. 165, pp. 525 – 531, 2019, 2nd International Conference on Recent Trends in Advanced

Computing ICRTAC -DISRUP - TIV INNOVATION , 2019 November 11-12, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050920300326

[24] C. library, [online], 2020. [Online]. Available: https://www.cryptopp.com/benchmarks.html

[25] Z. He, L. Wu, and X. Zhang, "High-speed pipeline design for hmac of sha-256 with masking scheme," in *2018 12th IEEE International Conference on Anti-counterfeiting, Security, and Identification (ASID)*, 2018, pp. 174–178.

[26] R. K. Santhosh, R. Shashidhar, A. M. Mahalingaswamy, M. S. K. Praveen, and M. Roopa, "Design of high speed aes system for efficient data encryption and decryption system using fpga," in *2018 International Conference on Electrical, Electronics, Communication, Computer, and Optimization Techniques (ICEECCOT)*, 2018, pp. 1279–1282.

Seznam symbolů, veličin a zkratek

ICS	Industrial Control System
KI	Kritická Infrastruktura
MW	mega Watt
kV	kilo Volt
PS	Phase Shifting
USA	United States of America
MS	Microsoft
VPN	Virtual Private Network
DoS	Denial of Service
HTTP	Hypertext Transfer Protocol
DDoS	Distributed Denial of Service
API	Aplication Programming Interface
IEC	International Electrotechnical Commission
OPC	Object Linking and Embedding for Process Control
OPC DA	OPC Data Access
GUID	Globally Unique Identifier
RTU	Remote Terminal Unit
DLL	Dynamic-Link Library
MMS	Manufacturing Message Specification
PLC	Programmable Logic Controller
RPC	Remote Procedure Call
HMI	Human-Machine Interface
SCADA	Supervisory Control and Data Acquisition
GOOSE	Generic Object Oriented Substation Event
SMV	Sampled Measured Values
IED	Inteligent Electronical Device
LD	Logical Device
LN	Logical Node
DO	Data Object
TLV	Type-Length-Value
VMD	Virtual Manufacturing Device
ISO	International Organisation for Standardization
TCP	Transmission Control Protocol
TPKT	ISO Transport Service on top of the TCP
COTP	Connection Oriented Transport Protocol
IP	Internet Protocol
IT	Informační Technologie

OT	Operační Technologie
PC	Personal Computer
RPi	Raspberry Pi
NTP	Network Time Protocol
MITM	Man In The Middle
MAC	Media Access Control
VLAN	Virtual Local Area Network
GHz	gigaherz
GB	gigabyte
MB	megabyte
B	byte
DHCP	Dynamic Host Configuration Protocol
ARP	Address Resolution Protocol
TLS	Transport Layer Security
TSAP	Transport Service Access Point
NAT	Network Address Translation
STL	Standard Template Library
PID	Process ID
RSA	Rivest, Shamir, Adleman
HMAC	Hash-Based Message Authentication Code
SHA	Secure Hash Algorithm
Mbps	Megabity za sekundu
DHE-RSA	Diffie-Hellman Exchange - Rivest-Shamir-Adleman
ECDHE-RSA	Elliptic-curve Diffie-Hellman - RSA
ECDHE-ECDSA	Elliptic Curve Digital Signature Algorithm
SDN	Software Defined Networking